一本书读懂Android外设开发

薛 伟 陈 强◎编著

Android外设开发实战

- 实录520分钟、91个外设开发高清学习视频。
- 11个大型综合案例，与实际外设开发项目可无缝对接。
- 从内核分析到接口实现，完整再现一个个经典外设项目的开发全程。
- 教授精髓，精讲精炼。赠送源码，拿来就用。



- 15个Android综合项目开发案例
- 38个Android应用开发学习视频

清华大学出版社

Android外设开发实战

薛　伟　陈　强　编著

清华大学出版社
北　京

内 容 简 介

Android 系统从诞生到现在，短短几年便凭借其操作易用性和开发的简洁性，赢得了广大用户和开发者的支持。截至 2014 年 9 月 30 日，Android 系统的市场占有率高达 85%。本书内容分为 3 篇，共计 19 章，循序渐进地讲解了开发 Android 外设项目的基本知识。本书从获取源码和搭建应用开发环境开始讲起，依次讲解了基础知识、系统分析和实战演练 3 部分的内容。在讲解每一个知识点时，都遵循了理论联系实际的讲解方式，从内核分析到接口 API 实现，再到实战演练，最后到综合实例演练，彻底剖析了一个个经典外设的完整实现流程。本书几乎涵盖了所有 Android 外设项目开发的主要内容，讲解通俗易懂并且详细，不但适合应用高手的学习，也特别有利于初学者学习和掌握。

本书适合 Android 驱动开发者、Linux 开发人员、Android 物联网开发人员、Android 编程爱好者、Android 源码分析人员、Android 应用开发人员、Android 传感器开发人员、Android 智能家居开发人员、Android 可穿戴设备开发人员的学习，也可以作为相关培训机构和大专院校相关专业的教学用书。

图书在版编目（CIP）数据

Android 外设开发实战/薛伟，陈强编著. —北京：清华大学出版社，2015
ISBN 978-7-302-40182-7

I. ①A… II. ①薛… ②陈… III. ①移动终端-应用程序-程序设计 IV. ①TN929.53

中国版本图书馆 CIP 数据核字（2015）第 101568 号

责任编辑： 朱英彪
封面设计： 刘　超
版式设计： 刘艳庆
责任校对： 王　颖
责任印制：

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn，http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 203mm×260mm　**印　　张：** 39.75　**字　　数：** 1085 千字
（附 DVD 光盘 1 张）
版　　次： 2015 年 7 月第 1 版　**印　　次：** 2015 年 7 月第 1 次印刷
印　　数： 1～3000
定　　价： 86.00 元

产品编号：061535-01

前　言

2007 年 11 月 5 日，谷歌公司宣布的基于 Linux 平台的开源手机操作系统 Android 诞生，该平台号称是首个为移动终端打造的真正开放和完整的移动软件。本书将和广大读者一起共同领略这款系统的神奇之处。

市场占有率高居第一

截至 2014 年 9 月，Android 在手机市场上的占有率从 2013 年的 68.8%上升到 85%。从数据上看，Android 市场的占有率增加幅度较大，WP 市场小幅增长，但 iOS 却有所下降。Android 平台占据了市场的主导地位，继续充当老大的角色。

就目前来看，智能手机的市场已经饱和，大多数人都在各个平台中转换。而就在这样一个市场上，Android 还增长了 10%左右的占有率确实不易。

为开发人员提供了成长的“沃土”

（1）保证开发人员可以迅速转型为 Android 应用开发

Android 应用程序是通过 Java 语言开发的，只要具备 Java 开发基础，就能很快地上手并掌握。作为单独的 Android 应用开发，对 Java 编程门槛的要求并不高，即使没有编程经验的生手，也可以在突击学习 Java 之后容易地学习 Android。

（2）定期召开奖金丰厚的 Android 大赛

为了吸引更多的用户使用 Android 开发，谷歌已经成功举办了奖金为数千万美元的开发者竞赛，鼓励开发人员研发出创意十足、十分有用的软件。

（3）开发人员可以利用自己的作品赚钱

为了能让 Android 平台吸引更多的关注，谷歌提供了一个专门下载 Android 应用的门店：Android Market，地址是 https://play.google.com/store。该门店允许开发人员发布应用程序，也允许 Android 用户下载自己喜欢的程序。作为开发者，需要申请开发者账号，申请后才能将自己的程序上传到 Android Market，并且可以对自己的软件进行定价。

本书的内容

本书内容分为 3 篇，共计 19 章，循序渐进地讲解了开发 Android 外设项目的基本知识。本书依次讲解了 Android 系统、获取并编译 Android 源码、搭建 Android 应用开发环境、Android 核心框架、Android 传感器系统架构、蓝牙系统、NFC 近场通信、Google Now 和 Android Wear、暴走轨迹计步器、智能家

居系统、智能心率计、湿度测试仪、小米录音机、智能楼宇灯光控制系统、智能闹钟系统、开发一个音乐播放器、移动阅读器系统、QR 码采集器和骑行记录仪等知识。

本书的版本

Android 系统自 2008 年 9 月发布第一个版本 1.1 至 2014 年 10 月发布最新版本 5.0，一共存在十多个版本。由此可见，Android 系统升级频率较快，一年之中最少有两个新版本诞生。但如果过于追求新版本，会造成力不从心的结果，所以在此建议广大读者："不必追求最新的版本，我们只需关注最流行的版本即可"。据官方统计，截至 2014 年 10 月 25 日，占据前 3 位的版本分别是 Android 4.3、Android 4.4 和 Android 4.2，其实这 3 个版本的区别并不是很大，只是在某领域的细节上进行了更新。为了及时体验 Android 系统的最新功能，本书中使用的版本是目前（本书成稿时）最新的版本 Android 5.0。

本书特色

本书内容十分丰富，并且讲解细致。我们的目标是通过一本图书，提供多本图书的价值，读者可以根据自己的需要有选择地阅读。在内容的编写上，本书具有以下特色。

（1）内容全面，讲解细致

本书几乎涵盖了开发 Android 项目所需要的所有主要知识点，详细讲解了每一个典型外设项目的实现过程和具体移植方法。每一个知识点都力求用详实和易懂的语言展现在读者面前。

（2）遵循合理的主线进行讲解

为了使广大读者彻底弄清楚 Android 外设项目开发的各个知识点，在讲解每一个知识点时，从 Linux 内核开始讲起，依次剖析了底层架构、API 接口连接和具体应用的知识。遵循了从底层到顶层，实现了外设项目开发大揭秘的目标。

（3）章节独立，自由阅读

本书中的每一章内容都可以独自成书，读者既可以按照本书编排的章节顺序进行学习，也可以根据自己的需求对某一章节进行针对性的学习，并且和传统古板的计算机书籍相比，阅读本书会带来很大的快乐。

（4）实例典型，实用性强

本书讲解了实际应用中最典型外设系统的实现方法和架构技巧，这些外设应用都是在商业项目中最需要的部分。读者可以直接将本书中介绍的知识应用到自己的项目中，实现无缝对接。

（5）附配资源丰富

本书配有丰富的学习资源，除源代码、PPT 之外，还实录了 91 个高清学习视频，既有实用的知识点讲解视频，也有详细的实例开发视频，全面、深入、细致地解析 Android 外设开发的方方面面。除此以外，本书额外赠送了 38 个 Android 应用开发学习视频，以及 15 个 Android 应用开发综合案例，包括仿小米录音机、音乐播放器、跟踪定位系统、仿陌陌交友系统、手势音乐播放器、智能家居系统、湿度测试仪、象棋游戏、抢滩登陆游戏、九宫格数独游戏、健康饮食系统、仓库管理系统、个人财务系统、仿去哪儿酒店预定系统、仿开心网客户端等。通过这些附配资源，读者的学习过程会更加方便、快捷。

读者对象

本书适合初学 Android 编程的自学者、Android 驱动开发者、Linux 开发人员、Android 物联网开发人员、Android 编程爱好者、Android 源码分析人员、Android 应用开发人员、Android 传感器开发人员、Android 智能家居开发人员、Android 可穿戴设备开发人员学习，也可以作为相关培训学校和大专院校相关专业的教学用书。

参与本书编写的人员还有周秀、付松柏、邓才兵、钟世礼、谭贞军、张加春、王教明、万春潮、郭慧玲、侯恩静、程娟、王文忠、陈强、何子夜、李天祥、周锐、朱桂英、张元亮、张韶青、秦丹枫。

本书在编写过程中，得到了清华大学出版社工作人员的大力支持，正是各位编辑的求实、耐心地付出，才能使得本书在较短的时间内出版。另外也十分感谢我们的家人，在写作本书的时候给予了巨大支持。由于本书编写团队成员水平有限，纰漏和不尽如人意之处在所难免，诚请读者提出意见或建议，以便修订并使之更臻完善。另外我们提供了售后支持网站（http://www.chubanbook.com/）和 QQ 群（192153124），读者朋友如有疑问可以在此提出，一定会得到满意的答复。

编　者

目　录

第 1 篇　基础知识篇

第 2 篇 系统分析篇



第 3 篇　实战演练篇



第 1 篇　基础知识篇

第 1 章　Android 系统介绍

第 2 章　获取并编译 Android 源码

第 3 章　搭建 Android 应用开发环境

第 4 章　Android 核心框架详解



第 1 章　Android 系统介绍

2007 年，Google 公司推出了一款无与伦比的移动智能设备系统——Android，这是一种建立在 Linux 基础之上的为手机、平板等移动设备提供的软件解决方案。截至 2014 年，Android 系统的占有率高达 85%，已经成为了当今最受欢迎的智能设备系统之一。本章将引领读者一起来了解 Android 系统的发展历程和背景，充分体验这款操作系统的成功之处。

1.1　纵览主流智能设备系统

知识点讲解：光盘:视频\知识点\第 1 章\纵览主流智能设备系统.avi

在当今市面中有很多智能设备系统，特别是在移动智能设备领域。在 Android 系统推出之前，塞班、苹果和微软在移动智能系统领域互不相让，三足鼎立之势日渐明朗。在本节的内容中，将一一讲解市面中主流的智能系统。

1.1.1　昨日王者——Symbian（塞班）

Symbian 作为昔日智能手机的王者，在 2005 年至 2010 年曾一度风行，人们手中拿的很多都是诺基亚的 Symbian 手机，N70—N73—N78—N97，诺基亚 N 系列曾经被称为“N=无限大”的手机。对硬件要求低，操作简单，省电，软件资源多，是 Symbian 系统手机的重要特点，如图 1-1 所示。

symbian OS

图 1-1　Symbian 系统

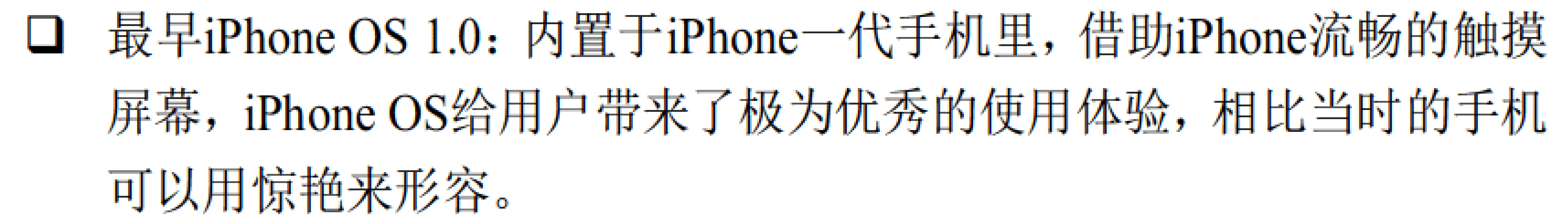
在国内软件开发市场内，基本每一个软件都会有对应的塞班手机版本。而塞班开发之初的目标是要保证在较低资源的设备上能长时间稳定可靠地运行，这导致了塞班的应用程序开发有着较为陡峭的学习曲线，开发成本较高，但是程序的运行效率很高。比如 5800 的 128MB 的 RAM，后台可以同时运行十几个程序而操作流畅（多任务功能是特别强大的），即使几天不关机它的剩余内存也能保持稳定。

虽然在 Android、iOS 的围攻之下，诺基亚推出了塞班 3 系统，甚至依然为其更新（Symbian Anna，Symbian Belle），从外在的用户界面到内在的功能特性都有了显著提升，例如可自由定制的全新窗体部件、更多主屏、全新下拉式菜单等。

由于对新兴的社交网络和 Web 2.0 内容支持欠佳，塞班占智能手机的市场份额日益萎缩。2010 年末，其市场占有量已被 Android 超过。自 2009 年底开始，包括摩托罗拉、三星电子、LG、索尼爱立信等各大厂商纷纷宣布终止塞班平台的研发，转而投入 Android 领域。2011 年初，诺基亚宣布将与微软成立战略联盟，推出基于 Windows Phone 的智能手机，从而在事实上放弃了经营多年的塞班，塞班退市已成定局。

1.1.2 高贵华丽——iOS

iOS 作为苹果移动设备 iPhone 和 iPad 的操作系统（见图 1-2），在 App Store 的推动之下，成为了世界上引领潮流的操作系统之一。原本这个系统名为 iPhone OS，直到 2010 年 6 月 7 日 WWDC 大会上宣布改名为 iOS。iOS 的用户界面的概念基础是能够使用多点触控直接操作。控制方法包括滑动、轻触开关及按键。与系统交互包括滑动（Swiping）、轻按（Tapping）、挤压（Pinching，通常用于缩小）及反向挤压（Reverse Pinching or Unpinching，通常用于放大）。此外，通过其自带的加速器，可以令其旋转设备改变其 y 轴，以令屏幕改变方向，这样的设计令 iPhone 更便于使用。



- 最早iPhone OS 1.0：内置于iPhone一代手机里，借助iPhone流畅的触摸屏幕，iPhone OS给用户带来了极为优秀的使用体验，相比当时的手机可以用惊艳来形容。



图 1-2 iOS 操作系统标志

- iPhone OS 2.0：随iPhone 3G发布，App Store诞生。App Store为第三方软件的提供者提供了方便而又高效的一个软件销售平台，在软件开发者与最终用户之间架起了一座沟通与销售的桥梁，从而极大地丰富了iPhone手机功能应用。
- iPhone OS 3.0：iPhone 3G开始支持复制粘贴。
- iOS 4：在iPhone 4推出的时候，苹果决定将原来iPhone OS系统重新定名为iOS，并发布新一代操作系统iOS 4。在这个版本里，开始正式支持多任务功能，通过双击HOME键实现。
- iOS 5：加入了Siri语音操作助手功能，用户可以与手机实现语言上的人机交互，该功能可以实现对用户的语音识别，完成一些较为复杂的操作，使用Siri来查询天气、进行导航、询问时间、设定闹钟、查询股票甚至发送短信等功能，方便了用户的使用。

从最初的 iPhone OS，演变至最新的 iOS 系统，iOS 成为了苹果新的移动设备操作系统，横跨 iPod Touch、iPad、iPhone，成为苹果最强大的操作系统，甚至新一代的 Mac OS X Lion 也借鉴了 iOS 系统的一些设计，可以说 iOS 是苹果的又一个成功的操作系统，给用户带来了极佳的使用体验。

由于其优秀的系统设计以及严格的 App Store，iOS 作为应用数量最多的移动设备操作系统，加上强大的硬件支持以及最新 iOS 5 内置的 Siri 语音助手，无疑使得用户体验得到了更大的提升，感受科技带来的好处。

1.1.3 全新面貌——Windows Phone

早在 2004 年时，微软就开始以“Photon”的计划代号开始研发 Windows Mobile 的一个重要版本更新。直到 2008 年，在 iOS 和 Android 的巨大冲击之下，微软重新组织了 Windows Mobile 的小组，并继续开发一个新的行动操作系统。

Windows Phone，简称 WP，是微软发布的一款手机操作系统（见图 1-3），它将微软旗下的 Xbox Live 游戏、Xbox Music 音乐与独特的视频体验集成到手机中。微软公司于 2010 年 10 月 11 日晚上 9 点 30 分正式发布了智能手机操作系统 Windows Phone，并将其使用接口称为“Modern”接口。2011 年 2 月，诺基亚与微软达成全球战略同盟并深度合作共同研发。2011 年 9 月 27 日，微软发布 Windows Phone 7.5。2012 年 6 月 21 日，微软正式发布 Windows Phone 8，采用和 Windows 8 相同的 Windows NT 内核，同时也针对市场的 Windows Phone 7.5 发布 Windows Phone 7.8。现有 Windows Phone 7 手机都将无法升级

至 Windows Phone 8。如图 1-4 所示为诺基亚 Windows Phone 手机界面。



图 1-3　Windows Phone 操作系统标志



图 1-4　Windows Phone 手机界面

Windows Phone 具有桌面定制、图标拖曳、滑动控制等一系列前卫的操作体验。其主屏幕通过提供类似仪表盘的体验来显示新的电子邮件、短信、未接来电、日历约会等，让人们对重要信息保持时刻更新。它还包括一个增强的触摸屏界面，更方便手指操作；以及一个最新版本的 IE Mobile 浏览器——该浏览器在一项由微软赞助的第三方调查研究中，和参与调研的其他浏览器和手机相比，可以执行指定任务的比例超过 48%。很容易看出微软在用户操作体验上所作出的努力，而史蒂夫·鲍尔默也表示："全新的 Windows 手机把网络、个人电脑和手机的优势集于一身，让人们可以随时随地享受到想要的体验。"

Windows Phone，力图打破人们与信息和应用之间的隔阂，提供适用于人们包括工作和娱乐在内完整生活的方方面面，是最优秀的端到端体验。

1.1.4　高端商务——BlackBerry OS（黑莓）

BlackBerry 系统，即黑莓系统（见图 1-5），是加拿大 Research In Motion（简称 RIM）公司推出的一种无线手持邮件解决终端设备的操作系统，由 RIM 自主开发。它和其他手机终端使用的 Symbian、Windows Mobile、iOS 等操作系统有所不同，BlackBerry 系统的加密性能更强，更安全。

BlackBerry™

图 1-5　BlackBerry 系统标志

安装有 BlackBerry 系统的黑莓机，不单单是指一台手机，而是由 RIM 公司所推出，包含服务器（邮件设定）、软件（操作接口）以及终端（手机）大类别的 Push Mail 实时电子邮件服务。

"黑莓"（BlackBerry）移动邮件设备基于双向寻呼技术。该设备与 RIM 公司的服务器相结合，依赖于特定的服务器软件和终端，兼容现有的无线数据链路，实现了遍及北美、随时随地收发电子邮件的梦想。这种装置并不以奇妙的图片和彩色屏幕夺人耳目，甚至不带发声器。"9·11"事件之后，由于 BlackBerry 及时传递了灾难现场的信息，而在美国掀起了拥有一部 BlackBerry 终端的热潮。

黑莓赖以成功的最重要原则——针对高级白领和企业人士，提供企业移动办公的一体化解决方案。企业有大量的信息需要即时处理，出差在外时，也需要一个无线的可移动的办公设备。企业只要装一



个移动网关，一个软件系统，用手机的平台实现无缝链接，无论何时何地，员工都可以用手机进行办公。它最大的方便之处是提供了邮件的推送功能：即由邮件服务器主动将收到的邮件推送到用户的手持设备上，而不需要用户频繁地连接网络查看是否有新邮件。

黑莓系统稳定性非常优秀，其独特定位也深得商务人士所青睐。可是也因此在大众市场上得不到优势，国内用户和应用资源也较少。

背景说明

（1）2010 年 9 月，诺基亚宣布将从 2011 年 4 月起从 Symbian 基金会（Symbian Foundation）手中收回 Symbian 操作系统控制权。由此看来，诺基亚在 2008 年全资收购塞班公司之后希望继续扩大塞班影响力的愿望并没有实现。

（2）在苹果和 Android 的强大市场攻势下，诺基亚在 2011 年 2 月 11 日宣布与微软达成广泛战略合作关系，并将 Windows Phone 作为其主要的智能手机操作系统。这家芬兰手机巨头试图通过结盟扭转颓势。

（3）2011 年 8 月 15 日，谷歌和摩托罗拉移动公司共同宣布，谷歌将以每股 40.00 美元现金收购摩托罗拉移动，总额约 125 亿美元，相比摩托罗拉移动股份的收盘价溢价了 63%，双方董事会都已全票通过该交易。谷歌 CEO 拉里·佩奇表示，摩托罗拉移动完全专注于 Android 系统，收购摩托罗拉移动之后，将增强整个 Android 生态系统。佩奇同时表示，Android 将继续开源，收购的一个目的是为了获得专利。

（4）2013 年 9 月 3 日，微软公司今日宣布将以 37.9 亿欧元的价格收购诺基亚的设备和服务部门，同时还将以 16.5 亿欧元的价格收购诺基亚的相关技术专利，本次交易总额达到 54.4 亿欧元，其中有 3.2 万名员工将从诺基亚转入微软，整笔交易预计将于 2014 年第一季度完成。

（5）2013 年 9 月 24 日消息，黑莓表示已经与由 Fairfax Financial Holdings 主导的财团达成交易，准备以 47 亿美元出售，但是后来没有任何爆炸性消息发布。

1.1.5　本书的主角——Android

Android 一词最早出现于法国作家利尔·亚当（Auguste Villiers de l'Isle-Adam）在 1886 年发表的科幻小说《未来夏娃》（*L'ève Future*）中，他将外表像人的机器起名为 Android，如图 1-6 所示。

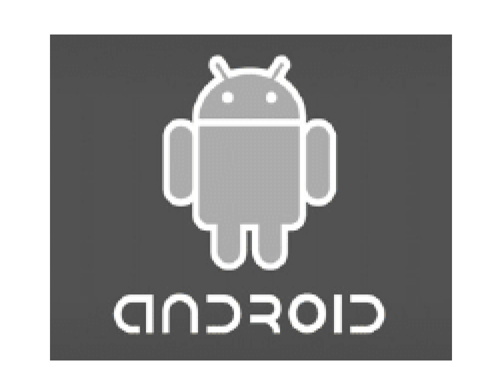


图 1-6　Android 标志

2008 年 HTC 和 Google 联手推出了第一台 Android 手机 G1，2014 年 10 月 15 日（美国太平洋时间），Google 公司发布了全新的 Android 操作系统：Android 5.0。北京时间 2014 年 6 月 26 日 0 时，Google I/O 2014 开发者大会在旧金山正式召开，发布了 Android 5.0 的前身 L（Lollipop）版 Android 开发者预览版本。2014 年的三款新 Nexus 设备——Nexus 6、Nexus 9 平板及 Nexus Player 将率先搭载 Android 5.0，之前的 Nexus 5、Nexus 7 及 Nexus 10 将会很快获得更新，而 Google Play 版设备则需要等上几周才能升级。

从 2011 年第一季度开始，Android 在全球的市场份额首次超过塞班系统，跃居全球第一。2014 年 8 月 15 日，根据 IDC 发布的 2014 年第二季度智能手机市场的最新数据显示，苹果 iOS 和谷歌 Android 两大系统平台继续领跑。Android 阵营增长则更惊人，达到了 33.3%，出货量达到了 2.553 亿台。Android 系统的市场份额得到了提高，从 2013 年第二季度的 79.6%增长到了 2014 年第二季度的 84.7%。具体信息如图 1-7 所示。

由此可见，Android 系统的市场占有率位居第一，并且毫无压力。Android 机型数量庞大，简单易用，相当自由的系统能让厂商和客户轻松地定制各种 ROM，以及各种桌面部件和主题风格。其简单而华丽的界面得到了广大客户的认可，对手机进行刷机也是不少 Android 用户所津津乐道的事情。

Top Five Smartphone Operating Systems, Worldwide Shipments, and Market Share, 2014Q2 (Units in Millions) - IDC/AppleInsider					
	Q2 2014 Shipment Volume	Q2 2014 Market Share	Q2 2013 Shipment Volume	Q2 2013 Market Share	Year-Over-Year Growth
Operating System					
Android	255.3	84.7%	191.5	79.6%	33.3%
iOS	35.2	11.7%	31.2	13.0%	12.7%
Windows Phone	7.4	2.5%	8.2	3.4%	-9.4%
BlackBerry	1.5	0.5%	6.7	2.8%	-78.0%
Others	1.9	0.6%	2.9	1.2%	-32.2%
Total	301.3	100.0%	240.5	100.0%	25.3%

图 1-7　2014 年 8 月智能手机平台调查表

可惜 Android 版本数量较多，市面上同时存在着 1.6、2.0、2.1、2.2、2.3、4.4.2、5.0 等各种版本的 Android 系统手机，应用软件对各版本系统的兼容性对程序开发人员是一个不小的挑战。同时由于开发门槛低，导致应用数量虽然很多，但是应用质量参差不齐，甚至出现不少恶意软件，导致一些用户受到损失。同时 Android 没有对各厂商在硬件上进行限制，导致一些用户在低端机型上体验不佳。另一方面，因为 Android 的应用主要使用 Java 语言开发，其运行效率和硬件消耗一直是其他手机用户所诟病的地方。

1.2　分析 Android 成功的秘诀

知识点讲解：光盘:视频\知识点\第 1 章\分析 Android 成功的秘诀.avi

从 2007 年诞生，到 2014 年占据市场 85%的份额，为什么 Android 系统能够在这么短的时间内成为移动智能设备市场占有率的第一名？在本节的内容中，将从 4 个方面来为读者解答这个问题。

1.2.1　强有力的业界支持

Android 系统基于 Linux 内核，是一款开源的手机操作系统。正是因为如此，在 Android 刚刚崭露头角之后，各大手机厂商和电信部门纷纷加入到了 Android 联盟当中。Android 联盟由业界内的世界级企业组成，主要成员包括中国移动、摩托罗拉、高通、T-Mobile、三星、LG、HTC 等在内的 30 多家技术和无线应用的领军企业。Android 通过与运营商、设备制造商、开发商和其他有关各方结成深层次的合作伙伴关系，希望借助建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。

1.2.2　研发阵容强大

Android 的研发队伍阵容强大，包括摩托罗拉、Google、HTC（宏达电子）、PHILIPS、T-Mobile、高通、魅族、三星、LG 以及中国移动在内的 34 家企业，这一个个响亮的名字都在业界内堪称大佬。他们都将基于该平台开发手机的新型业务，应用之间的通用性和互联性将在最大程度上得到保持。无论是从硬件到软件，还是到电信服务商，Android 从一开始便成为了业界内的宠儿，被当做重点新秀而培养。这样 Android 系统在强大的开发团队的培育和呵护下，最终顺利地功成名就，成为了一方霸主。

1.2.3　为开发人员“精心定制”

Google 公司一直视程序员为前进动力的源泉，为了提高程序员们的开发积极性，不但为开发人员提供了一流的开发装备和软件服务，而且还提出了振奋人心的奖励机制。

（1）保证开发人员可以迅速转型为 Android 应用开发

Android 应用程序是通过 Java 语言开发的，只要具备 Java 开发基础，就能很快地上手并掌握。作为单独的 Android 应用开发，对 Java 编程门槛的要求并不高，即使没有编程经验的门外汉，也可以在突击学习 Java 之后学习 Android。另外，Android 完全支持 2D、3D 和数据库，并且和浏览器实现了集成。所以通过 Android 平台，程序员可以迅速、高效地开发出绚丽多彩的应用，例如常见的工具、管理、互联网和游戏等。

（2）定期召开奖金丰厚的 Android 大赛

为了吸引更多的用户使用 Android 开发，Google 已经成功举办了奖金为数千万美元的开发者竞赛，鼓励开发人员研发出创意十足、十分有用的软件。这种大赛对于开发人员来说，不但能练习自己的开发水平，并且高额的奖金也是学员们学习的动力。

（3）开发人员可以利用自己的作品赚钱

谷歌提供了一个专门下载 Android 应用的门店：Android Market，地址是 https://play.google.com/store。在这个门店里面允许开发人员发布应用程序，也允许 Android 用户下载自己喜欢的程序。作为开发者，需要申请开发者账号，申请后才能将自己的程序上传到 Android Market，并且可以对自己的软件进行定价。只要你的软件程序足够吸引人，你就可以获得很好的金钱回报。这样实现了程序员学习和赚钱两不误，所以吸引了更多开发人员加入到 Android 大军中来。

1.2.4　开源

Android 是一款开源的系统，开源意味着对开发人员和手机厂商来说是完全无偿免费使用的。正是因为这一原因，所以吸引了全世界各地无数程序员的热情。于是很多手机厂商都纷纷采用 Android 作为自己产品的系统，这当然也包括很多山寨厂商。因为免费所以降低了成本，提高了利润。而对于开发人员，因为 Android 深受众多移动设备产品所采用，所以这方面的人才也变得愈发珍贵。正是因为开源，Android 系统也被广大非移动设备所采用，例如车载系统、智能电视、智能家居、穿戴设备等。

1.3　Android 智能设备来袭

知识点讲解：光盘:视频\知识点\第 1 章\Android 智能设备来袭.avi

因为 Android 系统的免费和开源，也因为系统本身强大的功能性，使得 Android 系统不仅被用于手机和平板设备上，而且也被广泛用于其他智能设备中，例如当前的新兴热点可穿戴设备。在本节的内容中，将简要介绍在市面中常见的搭载 Android 系统的智能设备。

1.3.1 常见的 Android 智能设备

（1）智能电视

Android 智能电视是指搭载了安卓操作系统的电视，具有智能化，能实现网页浏览、视频电影观看、聊天办公游戏等，与平板电脑和智能手机一样的功能。其凭借安卓系统让电视实现智能化的提升，数十万款安卓市场的应用、游戏等内容随意安装。例如海尔的模卡（MOOKA）U42H7030 便是一款搭载 Android 4.2 系统的智能电视，如图 1-8 所示。

（2）机顶盒

Android 机顶盒是指像智能手机一样，具有全开放式平台，搭载了安卓操作系统，可以由用户自行安装和卸载软件、游戏等第三方服务商提供的程序，通过此类程序不断对电视的功能进行扩充，并可以通过网线、无线网络实现上网冲浪的新一代机顶盒总称。

通过使用 Android 机顶盒，可以让电视具有上网、看网络视频、玩游戏、看电子书、听音乐等功能，使电视成为一个低成本的平板电脑。Android 机顶盒，不仅仅是一个高清播放器，更具有一种全新的人机交互模式，既区别于电脑，又有别于触摸屏，Android 机顶盒配备红外感应条，遥控器一般采用空中飞鼠，这样就可以方便地实现触摸屏上的各种单点操作，可以方便地在电视上玩愤怒的小鸟、植物大战僵尸等经典游戏。例如乐视公司的 Letv 机顶盒便是基于 Android 打造的，如图 1-9 所示。



图 1-8 搭载 Android 4.2 系统的智能电视



图 1-9 基于 Android 的 Letv 机顶盒

（3）游戏机

Android 游戏机就像 Android 智能手表一样，在 2013 年出现了爆炸式增长。在 CES 展会上，NVIDIA 的 Project Shield 掌上游戏主机以绝对震撼的姿态亮相，之后又有 Ouya 和 Gamestick 相继推出。不久前，Mad Catz 也发布了一款 Android 游戏机。

（4）智能手表

智能手表，是将手表内置智能化系统、搭载智能手机系统而连接于网络实现多功能，能同步手机中的电话、短信、邮件、照片、音乐等。2013 年 3 月媒体报道，苹果、三星、谷歌等科技巨头都将在 2013 年晚些时候发布智能手表。美国市场研究公司 Current Analysis 分析师艾维·格林加特（Avi Greengart）认为 2013 年可能会成为智能手表元年。例如 LG 宣布采用谷歌 Android Wear 操作系统开发了一款名为 G Watch 的智能手表，该产品于 2014 年第二季度发布，如图 1-10 所示。

（5）智能家居

智能家居是以住宅为平台，利用综合布线技术、网络通信技术、智能家居-系统设计方案安全防范技术、自动控制技术、音视频技术将家居生活有关的设施集成，构建高效的住宅设施与家庭日程事务

的管理系统，提升家居安全性、便利性、舒适性、艺术性，并实现环保节能的居住环境。

智能家居是在互联网的影响之下的物联化体现。智能家居通过物联网技术将家中的各种设备（如音视频设备、照明系统、窗帘控制、空调控制、安防系统、数字影院系统、网络家电以及三表抄送等）连接到一起，提供家电控制、照明控制、窗帘控制、电话远程控制、室内外遥控、防盗报警、环境监测、暖通控制、红外转发以及可编程定时控制等多种功能和手段。与普通家居相比，智能家居不仅具有传统的居住功能，还兼备建筑、网络通信、信息家电、设备自动化，集系统、结构、服务、管理为一体的高效、舒适、安全、便利、环保的居住环境，提供全方位的信息交互功能，帮助家庭与外部保持信息交流畅通，优化人们的生活方式，帮助人们有效安排时间，增强家居生活的安全性，甚至为各种能源费用节约资金。

例如乐得威公司的 GW-9311 智能主机产品便是一款 Android 智能家居产品，如图 1-11 所示。



图 1-10　搭载 Android Wear 系统的 G Watch

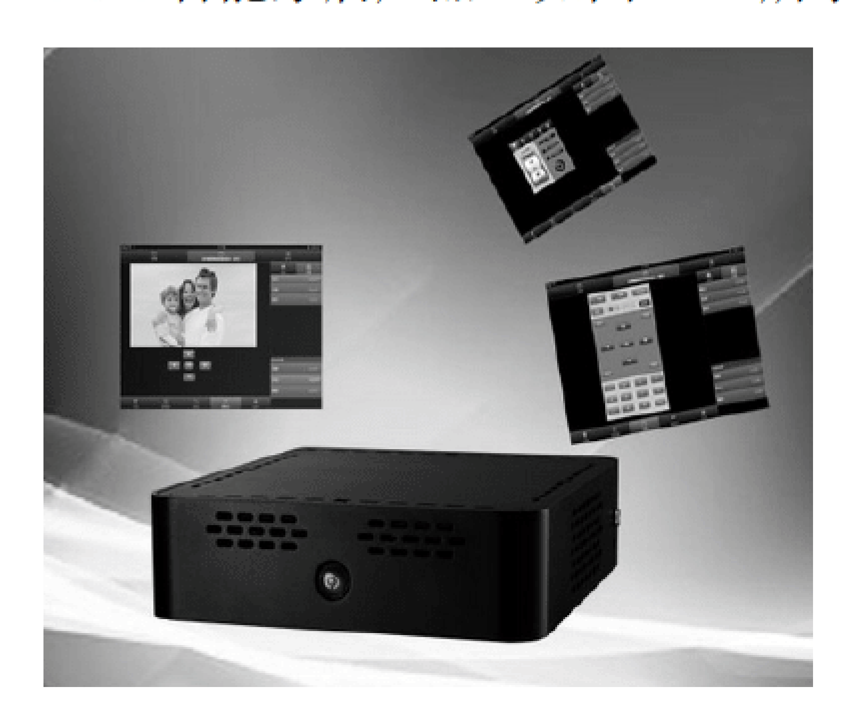


图 1-11　乐得威公司的 GW-9311 智能主机

上述智能设备只是冰山一角，随着物联网和云服务的普及和发展，将会有更多的智能设备诞生，Android 系统将拥有一个更加美好的未来。

1.3.2　新兴热点——可穿戴设备

最近两年，随着 Android 和 iOS 系统的发展，可穿戴设备逐渐展现在广大用户的面前。谷歌眼镜、苹果手表等新颖而又时尚的设备吸引了广大用户的眼球，相信在未来这些设备必将引领时尚的潮流，成为科技界的主流产品之一。自从谷歌推出 Google 眼镜产品之后，可穿戴计算设备便成为了当今科技界的火热话题之一。在 CES 2013 和 CES 2014（国际电子展）上，也有不少公司推出了眼镜、腕带等各种可穿戴计算设备，可穿戴计算也越来越火热。

1. 发展背景

穿戴设备看似是一个新兴事物，但实际上其发展历史可以上溯到 20 世纪 80 年代。多伦多大学教授 Steve Mann 被人称为“可穿戴计算之父”，公认的第一个赛博格（Cyborg）——这是一个特殊的群体，他们很像科幻小说中的一些角色，利用机器设备来增强自己的感觉，从而加强对环境的掌控。

Steve Mann 自 20 世纪 80 年代就开始尝试制作类似于 Google Glass 这样可以架在自己的鼻梁上，以第一人称的角度来记录周遭事物的眼镜。Mann 最初设计的设备是戴在头盔上的，而经过多年的实验和反复改进，他的头戴式智能眼镜变得越来越轻巧。后来 Mann 成功开发出令智能眼镜小型化，并与电脑和网络相连的技术 EyeTab，这比 Google 眼镜要早 13 年。

以 Google 眼镜为代表的这种穿戴设备，和智能手机最大的不同是把用户的眼睛和手从设备上解放

出来了，所以不需要从兜里掏出一个东西，也不需要低下头去看它，它永远在你前面。所以我们有理由相信，可穿戴计算设备（不一定是 Google 眼镜）一定可以给人们带来更大的自由，并且在将来一定会成为潮流趋势。

2. 发展现状介绍

可穿戴计算设备将成为继智能手机、平板电脑之后的又一个潮流。当前可穿戴技术正处于一种过渡时期，一些看似疯狂的想法逐渐在摸索中变得更加成熟。在 CES 2014 电子消费展上，我们已经看到类似 Pebble Steel 这样拥有更精致设计的智能手表，还有很多其他运动腕带、智能眼镜等产品参与展出，下面一起来回顾一下这些出色的可穿戴设备。

（1）Google Project Glass

谷歌眼镜（Google Project Glass）是由谷歌公司于 2012 年 4 月发布的一款“拓展现实”眼镜，如图 1-12 所示。谷歌眼镜具有和智能手机一样的功能，可以通过声音控制拍照、视频通话和辨明方向以及上网冲浪、处理文字信息和电子邮件等。

2013 年 4 月 10 日，美国科技博客 Gizmodo 发布了一张图片，揭示了谷歌智能眼镜的工作原理。谷歌眼镜承载着可穿戴设备的开端，极具想象空间并且前途不可限量。但现在看来，其暂时只是一个手机伴侣，基础通信、文字输入依赖手机。

2013 年 11 月 12 日，谷歌发布了谷歌眼镜的一系列新功能，包括搜索歌曲、扫描已保存播放列表，以及收听高保真音乐等。美国东部时间 2014 年 4 月 15 日早上 9 点，Google Glass 正式开放网上订购。

（2）苹果智能手表

苹果智能手表，是苹果秘密研发的产品。苹果手表可能将采用 1.5 至 2 英寸显示屏，并将采用指纹识别技术。苹果成立了一支 100 人左右的开发团队，专门开发这款设备。2013 年 5 月，凯基证券分析师郭明池（Ming-Chi Kuo）在一份报告中指出，苹果手表的零部件尚未成熟，苹果手表最早 2014 年下半年投产。而苹果 CEO 蒂姆・库克（Tim Cook）曾表示，苹果期待 2013 年秋季和整个 2014 年将推出令人兴奋的新产品。苹果手表的预期效果如图 1-13 所示。

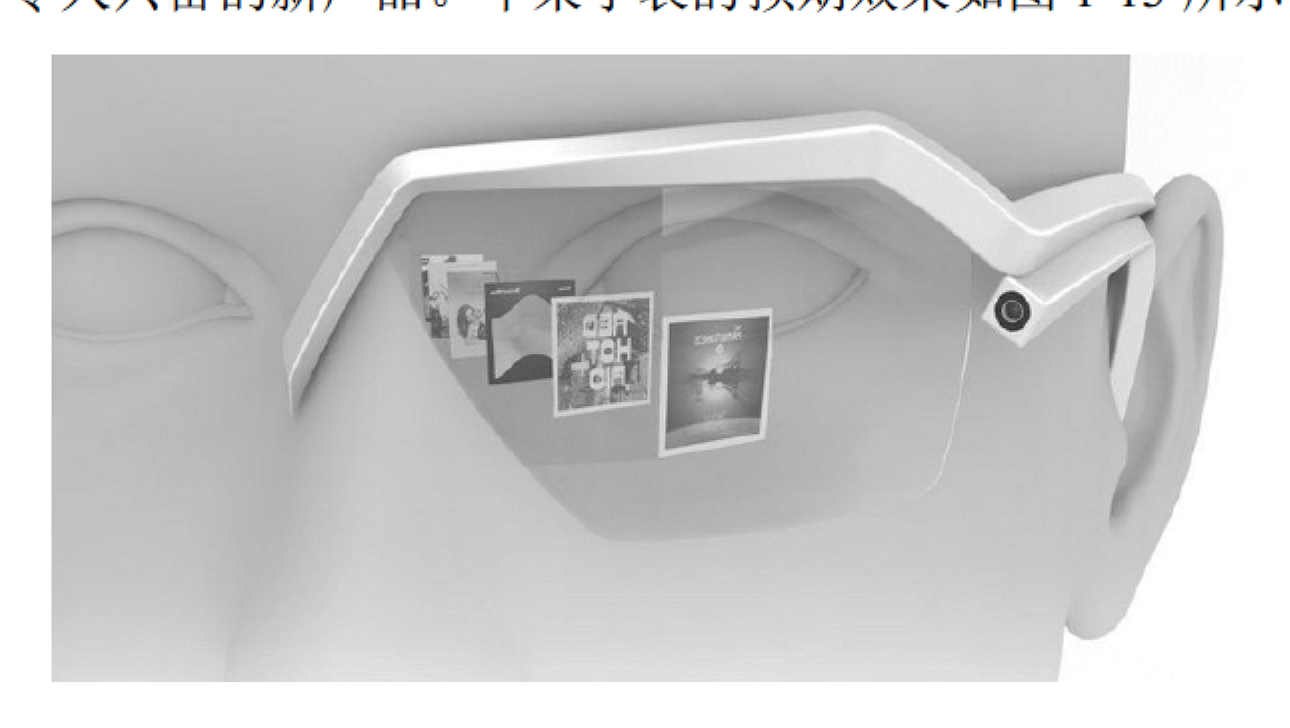


图 1-12　谷歌眼镜



图 1-13　苹果手表

（3）MetaWatch

MetaWatch 是一款智能手表，其设计师此前设计了奢侈品手机 Vertu，所以非常擅长将时尚元素与电子产品有机结合，如图 1-14 所示。

由此可以看到，MetaWatch 的金属表盘充满质感，皮革腕带也显得十分高级，无论是搭配西装还是 T 恤，都十分合适。MetaWatch 支持 10～15 米防水，支持 iOS 设备，用户可以从 App Store 中下载应用程序，实现更多信息的通知功能。

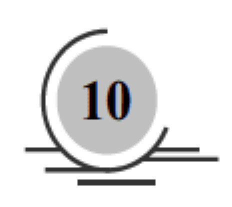




图 1-14　MetaWatch

（4）Garmin Vivofit

健身腕带 Garmin Vivofit 是由知名 GPS 厂商 Garmin 推出的，此前推出过运动手表产品，此次更是推出了 Vivofit 健身腕带，全面进入到运动监测设备市场。这款运动腕带的设计充满活力，拥有多种配色款式，不仅能够实现全面的运动数据监测，还支持心率监测，与手机端的应用搭配，可实现出色的健身运动功能，如图 1-15 所示。

（5）英特尔智能手表及手镯

芯片巨头英特尔在 CES 2014 上也宣布进入可穿戴设备领域，将陆续推出智能手表、手镯等产品。有意思的是，英特尔的智能手镯将与时尚百货 Barneys 合作推出，或许能够让可穿戴设备在时尚领域更进一步，如图 1-16 所示。



图 1-15　Garmin Vivofit 健身腕带



图 1-16　英特尔智能手表及手镯

1.3.3　可穿戴设备的发展前景分析

可穿戴设备是延续性地穿戴在人体上，具备先进的电路系统、无线联网及独立处理能力的终端设备，其具备最重要的两个特点是可长期穿戴和智能化。在智能手机和平板进入停滞期后，以智能眼镜、手表等为代表的智能可穿戴设备成为谷歌、苹果等巨头竞争的下一个主战场。著名科技媒体 Android Authority 的撰稿人奈特·斯旺纳（Nate Swanner）曾经撰文对 2014 年进行了展望，他总结道：穿戴设备将会蓬勃发展。消费者有望买到谷歌眼镜，而且市场上会出现很多智能手表。如果你有意购买一款这样的设备，请务必保持谨慎乐观的心态。毕竟，在这类产品爆炸式发展初期买下的设备，有可能在未来几年里登上“有史以来最糟糕产品”的榜单。

（1）智能手机推动力

智能手机是可穿戴设备爆发的核心驱动力之一，智能手机现在已经不光是拿来讲电话或者是上网，其内建的处理器与操作系统具有强大的运算能力，使它成为远程的计算机引擎，而同时智能手机具有广泛的用户基础，预计未来，手机可能作为智能控制中心和计算系统，使可穿戴设备、平板、笔记本、电视等所有终端保持互联，而每个人的身体及可穿戴设备将变成微网络，身上佩戴各式与智能手机连

接的装备，提供各类功能并与智能手机、云端进行数据计算和交互。根据权威统计数据证明，中国移动互联网用户已超过桌面互联网用户。

（2）跨国公司推动力

随着智能手机渗透率快速提升，便携性要求出现、硬件配备提升、传感器及电池改善，可穿戴设备的便携、云端互联等性能优势将越来越明显，预计可穿戴设备将是继智能手机之后下一个爆发性增长点。尤其是苹果、谷歌、微软、亚马逊和 Facebook 五大平台及相应开发者都进入可穿戴设备领域时，后台数据及前端检测传输更加完善时，可穿戴设备将会变成主流。按照 ShareThis 2013 年 6 月最新数据，消费者在移动设备上点击和分享内容的行为是桌面电脑的 2 倍，随着社交网络越发重要，可供分享的数据暴增。以社交分享平台中份额最高的 Facebook 和 Google 来研究，按照 Searchmetrics 数据，目前 Facebook 的分享量以每个月 10%的速度增长，Google 分享量目前以每月 19%的速度增长，截至 2013 年 4 月，Facebook 和 Google 的数据分享量相比 2012 年初增长分别是 202%及 788%，数据分享爆发时代到来。即时的数据分享和社交网络的需求将导致用户对各类移动终端需求不断提升，可穿戴设备在数据分享和社交领域具备放量基础。

（3）用户推动力

用户对健身、医疗及健康监测等需求也在持续抬升，可穿戴式设备在医疗和健康领域可加载的功能包括脉搏血氧仪、葡萄糖监测、心电图（ECG）、助听器、药物输送等。未来可穿戴设备作为新一代智能终端，将成为新的移动平台市场及生态圈，硬件终端不仅是营收增长点，也将成为粘住客户的产品形态，进而围绕消费者形成可穿戴设备、手机、平板、笔记本、电视、汽车等终端互联互通的一体化智能方案，因此原软硬件、互联网等各类厂商均参与到推出硬件终端产品的环节中来。

在可穿戴设备领域应用中，目前较受欢迎的应用是娱乐和社交，而较快进入商用的功能是健身、医疗及健康监测。娱乐和社交领域的典型产品包括 Google/百度智能眼镜、索尼/三星/果壳智能手表等，医疗领域可穿戴式设备主要包括脉搏血氧仪、葡萄糖监测、心电图（ECG）、助听器、药物输送等类型产品，目前主要功能进行一体化整合成为趋势，如三星智能手表同时也具备健康监测功能。

据数据显示，2012 年中国可穿戴便携移动医疗设备市场销售规模达到 4.2 亿元，预计到 2015 年这一市场规模将超过 10 亿元，到 2017 年中国可穿戴便携移动医疗设备市场销售规模将接近 50 亿元，市场年复合增长达到 60%。

HIS 预计全球范围内与健康相关的可穿戴设备 App 应用装机量（或下载量）会从 2012 年的 1.56 亿上升至 2017 年的 2.48 亿，随着开发者加入及生态环境改善，可穿戴设备将放量增长。

第三方机构 Endpoint Technologies Associates 预计，如果未来 5 年可穿戴设备市场占比达 4000 万个，便可能为开发商带来 4 亿美元商机，而程序内广告（in-APP advertising）可能大幅提升营业收入。

1.3.4 Android 对穿戴设备的支持——Android Wear

Bluetooth Smart 低耗能技术的推出为穿戴设备的开发创造了良好的条件。苹果是从 iOS 5（iPhone 4S 以及以上版本的手机）开始支持 Bluetooth Smart，而 Google 直到 Android 4.3 才开始支持。Google 在 Android 4.3 中添加了 Bluetooth Smart，在操作系统层面建立一个统一标准。也就是说，从 Android 4.3 开始，将完全支持新蓝牙传输技术，这样就为可穿戴设备开发铺平了道路。而在 Android 4.4 中，新增加了地磁旋转矢量、脚步探测器和计步器 3 个传感器类别，这些功能很可能是面向谣传的谷歌 Android 智能手表、谷歌眼镜以及非谷歌出厂的设备。随着更多的厂商在产品中加入运动传感器，追踪人们运

动的 Android 手机应用也将从该新功能中获益。

北京时间 2014 年 3 月 19 日早间消息，谷歌在官方博客中公布了可穿戴设备操作系统 Android Wear 的细节。Android Wear 是 Android 的一个修改版，基于 Google Now 语音识别技术，针对可穿戴计算设备设计，最初将被用在智能手表中。谷歌同时表示，LG、华硕、HTC、摩托罗拉移动和三星将是 Android Wear 的硬件合作伙伴，而博通、Imagination、英特尔、联发科和高通将是芯片合作伙伴。LG 和谷歌将在谷歌 I/O 开发者大会上发布智能手表，而 LG 将是推出谷歌智能手表的首家合作伙伴。

Android Wear 与谷歌眼镜类似，将基于 Google Now 和语音命令。通过 OK Google 的语音指令，用户可以提问或发送文字消息。谷歌表示，Android Wear 的设计是为了提供相关性更好的信息，以及来自社交媒体应用的通知、消息应用的提示，以及购物、新闻和拍照应用的通知等。这一修改版 Android 系统将专注于健康和运动追踪功能。FitBit Force 和耐克 FuelBand 等产品推动了这类功能的发展。谷歌还希望，Android Wear 将成为联系用户与其他设备，包括电视机和计算机的纽带。业内人士希望，谷歌的进入将有助于提升智能手表的设计美学。尽管可穿戴计算设备目前非常热门，但相关产品的销售情况并不火爆。三星第一代 Galaxy Gear 和索尼 SmartWatch 仍是小众产品，这两款产品的尺寸过大且不时尚。对可穿戴计算设备的其他不满还包括电池续航时间过短，以及缺少某些实用功能等。

第 2 章　获取并编译 Android 源码

Android 作为一项新兴技术，在进行开发前首先要搭建一个对应的开发环境。Android 开发包括底层开发和应用开发，底层开发大多数是指和硬件相关的开发，并且是基于 Linux 环境，例如开发驱动程序。因为本书讲解的外设项目开发涉及底层和应用层的知识，所以要求读者熟练运用底层开发环境和应用层开发环境。在本章的内容中，首先将详细讲解搭建 Android 底层开发环境的知识，详细介绍获取并编译 Android 源码的具体方法和实现流程。

2.1　在 Linux 系统中获取 Android 源码

知识点讲解：光盘:视频\知识点\第 2 章\在 Linux 系统中获取 Android 源码.avi

北京时间 2014 年 11 月 5 日，谷歌在 http://android.google.source.com/上正式发布了 Android 5.0 的源码，如图 2-1 所示。

根据图 2-1 所示的分支名可以下载 Android 5.0 的源码。在 Linux 系统中，通常使用 Ubuntu 来下载和编译 Android 源码。由于 Android 的源码内容很多，Google 采用了 git 的版本控制工具，并对不同的模块设置不同的 git 服务器，我们可以用 repo 自动化脚本来下载 Android 源码，下面介绍如何一步一步地获取 Android 源码的过程。

（1）下载 repo

在用户目录下，创建 bin 文件夹，用于存放 repo，并把该路径设置到环境变量中，命令如下：

```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
```

下载 repo 的脚本，用于执行 repo，命令如下：

```
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo
```

设置可执行权限，命令如下：

```
$ chmod a+x ~/bin/repo
```

（2）初始化一个 repo 的客户端

在用户目录下，创建一个空目录，用于存放 Android 源码，命令如下：

```
$ mkdir AndroidCode
$ cd AndroidCode
```

进入到 AndroidCode 目录，并运行 repo 下载源码，下载主线分支的代码，主线分支包括最新修改的 bug，以及并未正式发出版本的最新源码，命令如下：

```
$ repo init -u https://android.googlesource.com/platform/manifest
```

下载其他分支，正式发布的版本，可以通过添加-b 参数来下载，命令如下：

```
$ repo init -u https://android.googlesource.com/platform/manifest -b
                                        android-5.0_r1
```

在下载过程中会需要填写 Name 和 Email，填写完毕之后，选择 Y 进行确认，最后提示 repo 初始

化完成，这时可以开始同步 Android 源码了，同步过程很漫长，需要耐心地等待，执行下面命令开始同步代码：

```
$ repo sync
```

经过上述步骤后，便开始下载并同步 Android 源码了，界面效果如图 2-2 所示。

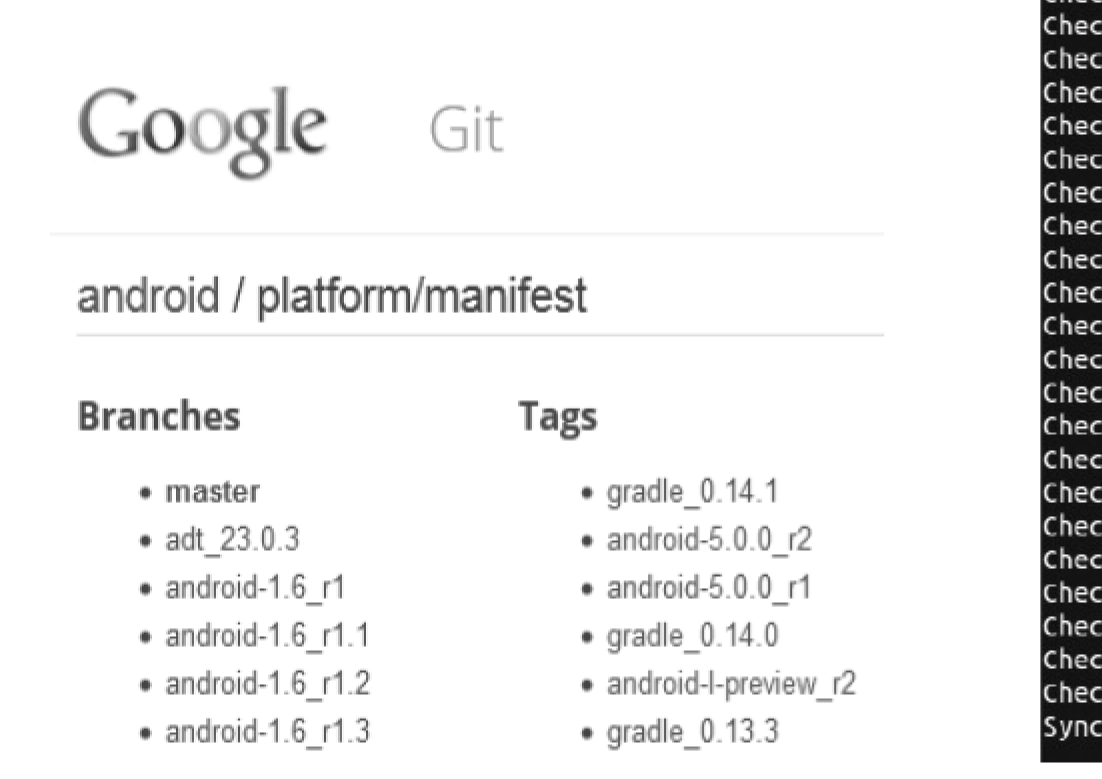


图 2-1　Android 5.0 的源码分支

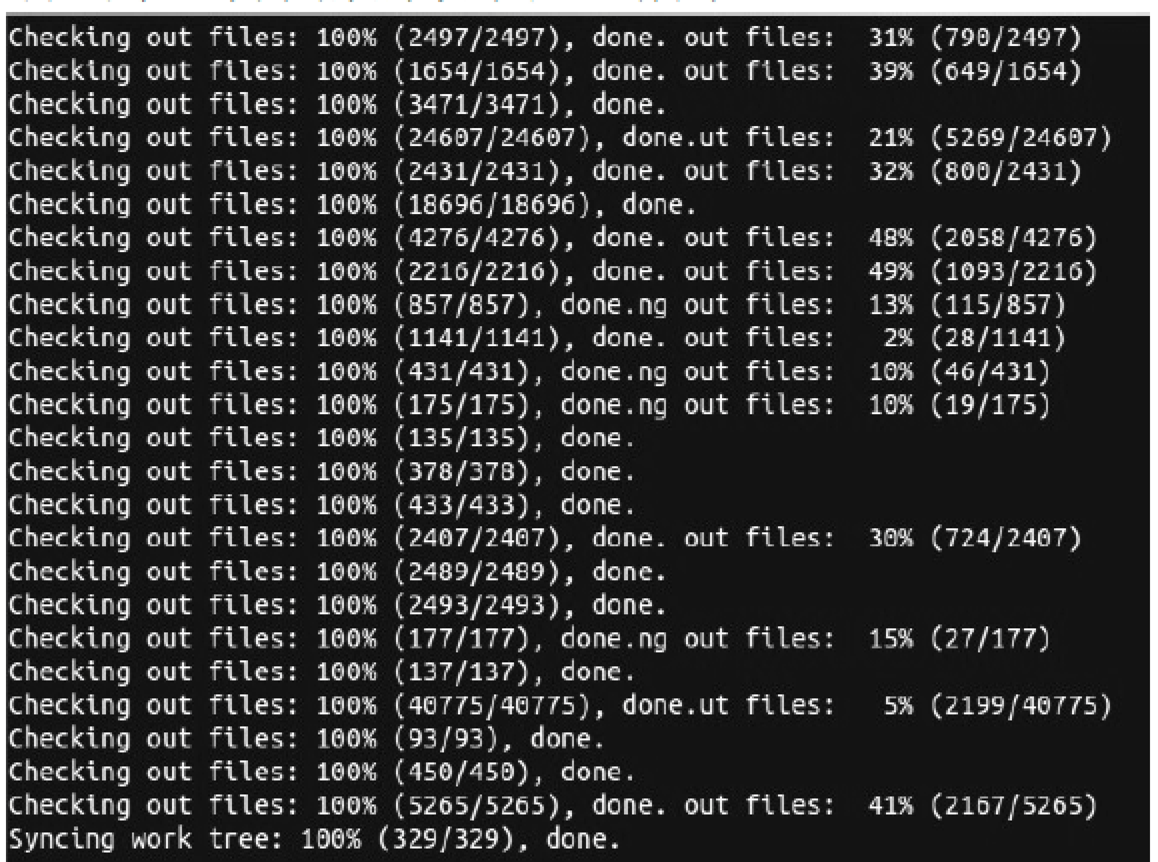


图 2-2　下载同步

2.2　在 Windows 平台获取 Android 源码

知识点讲解：光盘:视频\知识点\第 2 章\在 Windows 平台获取 Android 源码.avi

在 Windows 平台获取 Android 源码的原理和 Linux 相同，但是需要预先在 Windows 平台上面搭建一个 Linux 模拟环境，笔者使用的是 cygwin 工具。cygwin 的作用是构建一套在 Windows 上的 Linux 模拟环境，下载 cygwin 工具的地址如下：

```
http://cygwin.com/install.html
```

下载成功后会得到一个名为 setup.exe 的可执行文件，通过此文件可以更新和下载最新的工具版本，具体流程如下。

（1）启动 cygwin，如图 2-3 所示。



图 2-3　启动 cygwin

（2）单击“下一步”按钮，在进入的界面中选中第一个单选按钮，表示从网络下载安装，如图 2-4 所示。

（3）单击“下一步”按钮，在进入的界面中选择安装根目录，如图 2-5 所示。

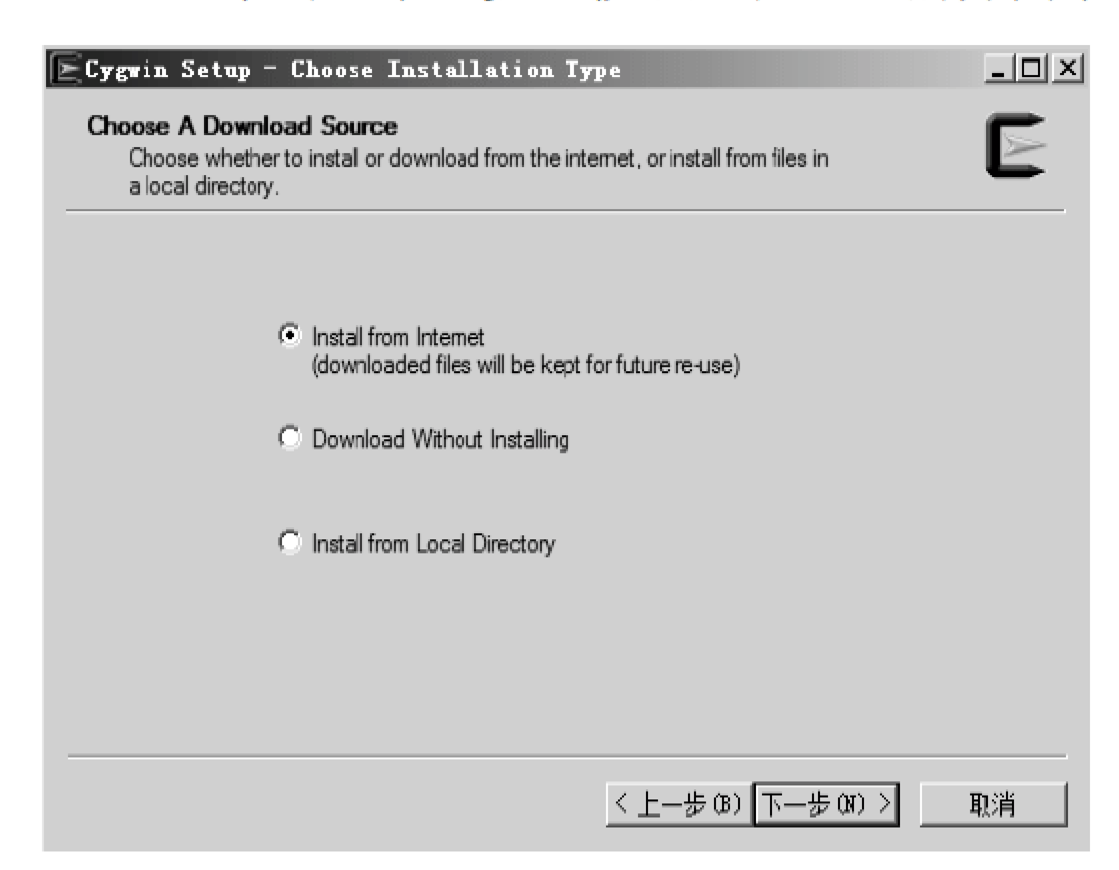


图 2-4　选择从网络下载安装

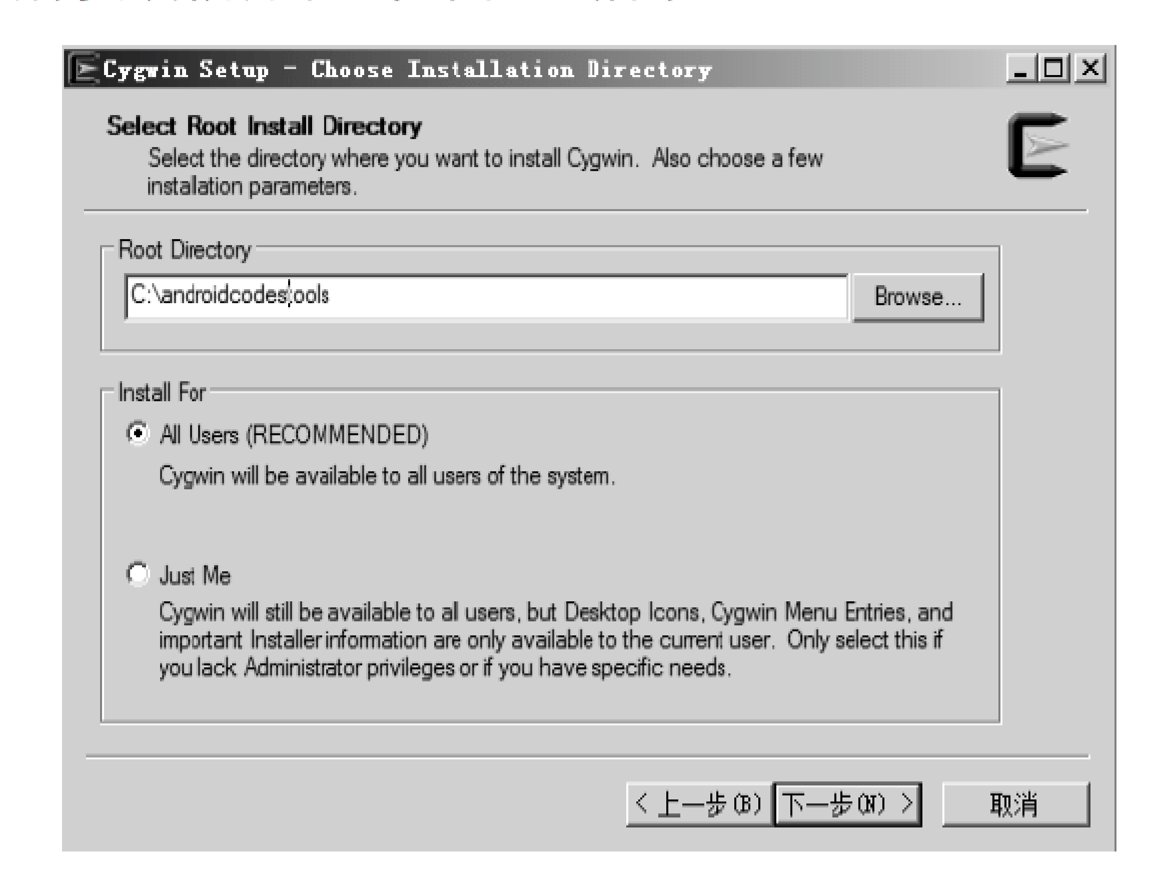


图 2-5　选择安装根目录

（4）单击“下一步”按钮，选择临时文件目录，如图 2-6 所示。

（5）单击“下一步”按钮，在进入的界面中设置网络代理。如果所在网络需要代理，则在这一步中进行设置，如果不用代理，则选择直接下载，如图 2-7 所示。

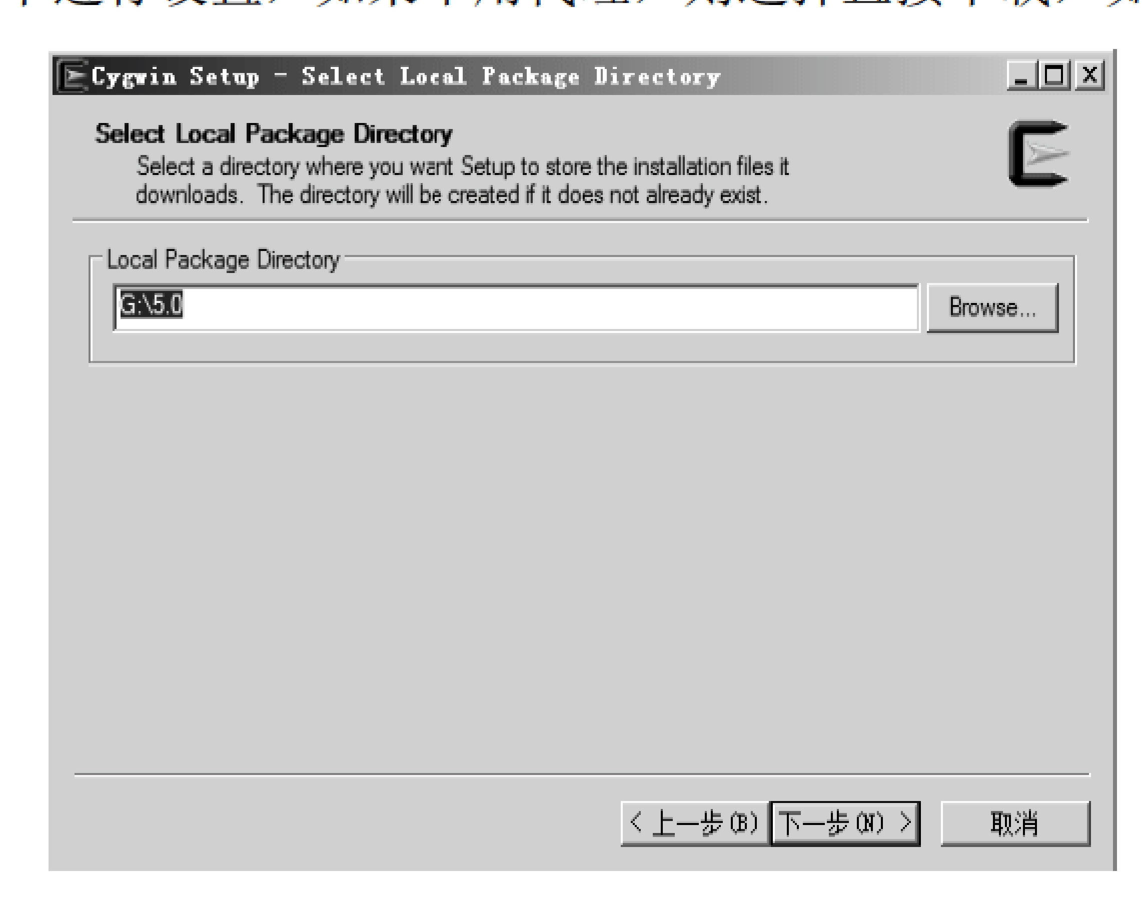


图 2-6　选择临时文件目录

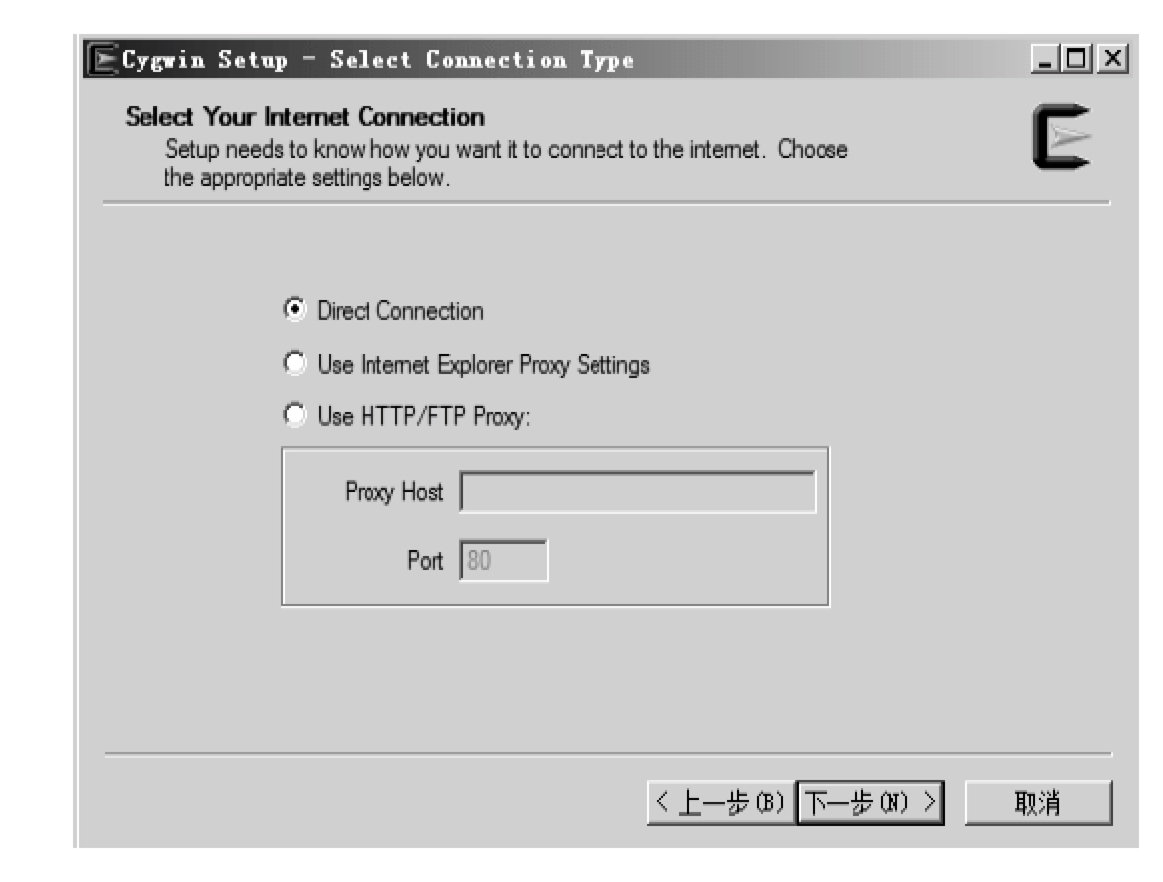


图 2-7　设置网络代理

（6）单击“下一步”按钮，在进入的界面中选择下载站点。一般选择离我们比较近的站点，速度会比较快，这里选择的是台湾站点，如图 2-8 所示。

（7）单击“下一步”按钮，开始更新工具列表，如图 2-9 所示。

（8）单击“下一步”按钮，在进入的界面中选择需要下载的工具包。在此我们需要依次下载 curl、git、python 这些工具，如图 2-10 所示。

为了确保能够安装上述工具，一定要用鼠标双击变为 Install 形式，如图 2-11 所示。

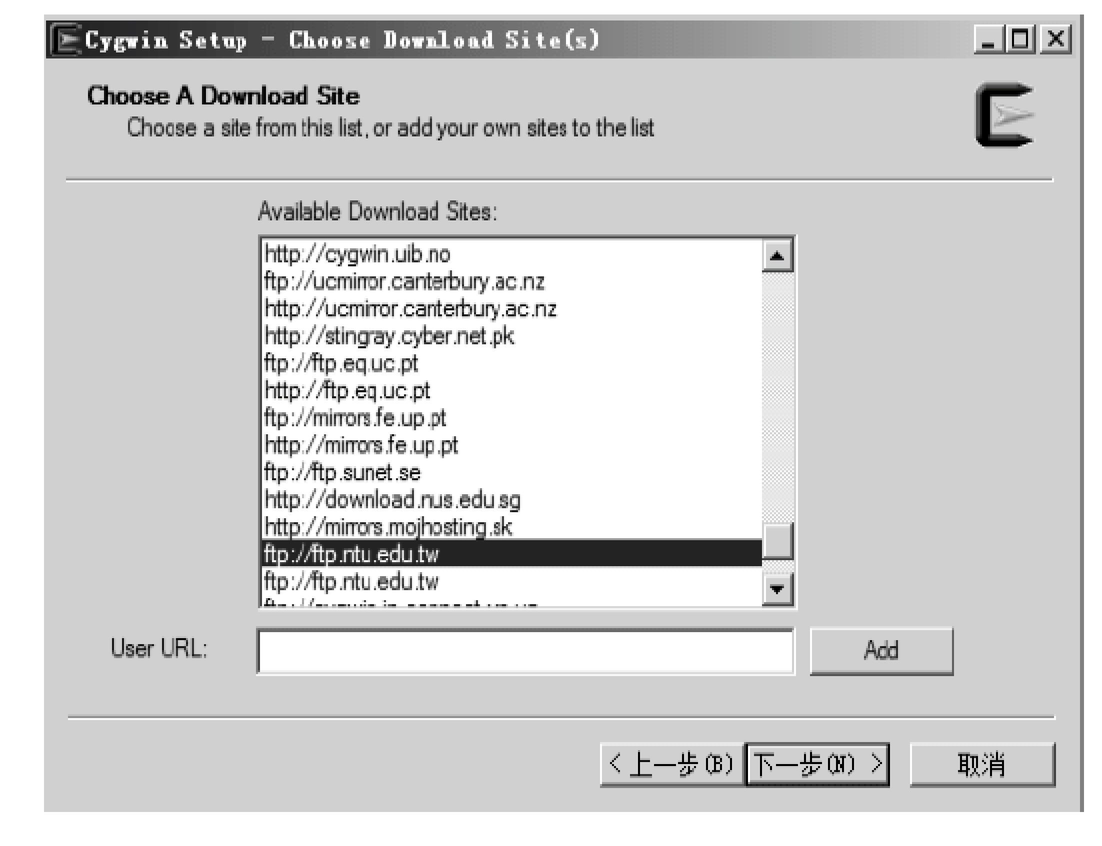
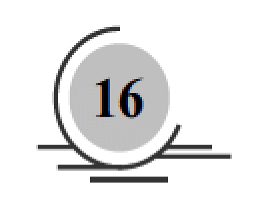


图 2-8　选择下载站点



（9）单击“下一步”按钮，进入漫长的等待过程，如图 2-12 所示。

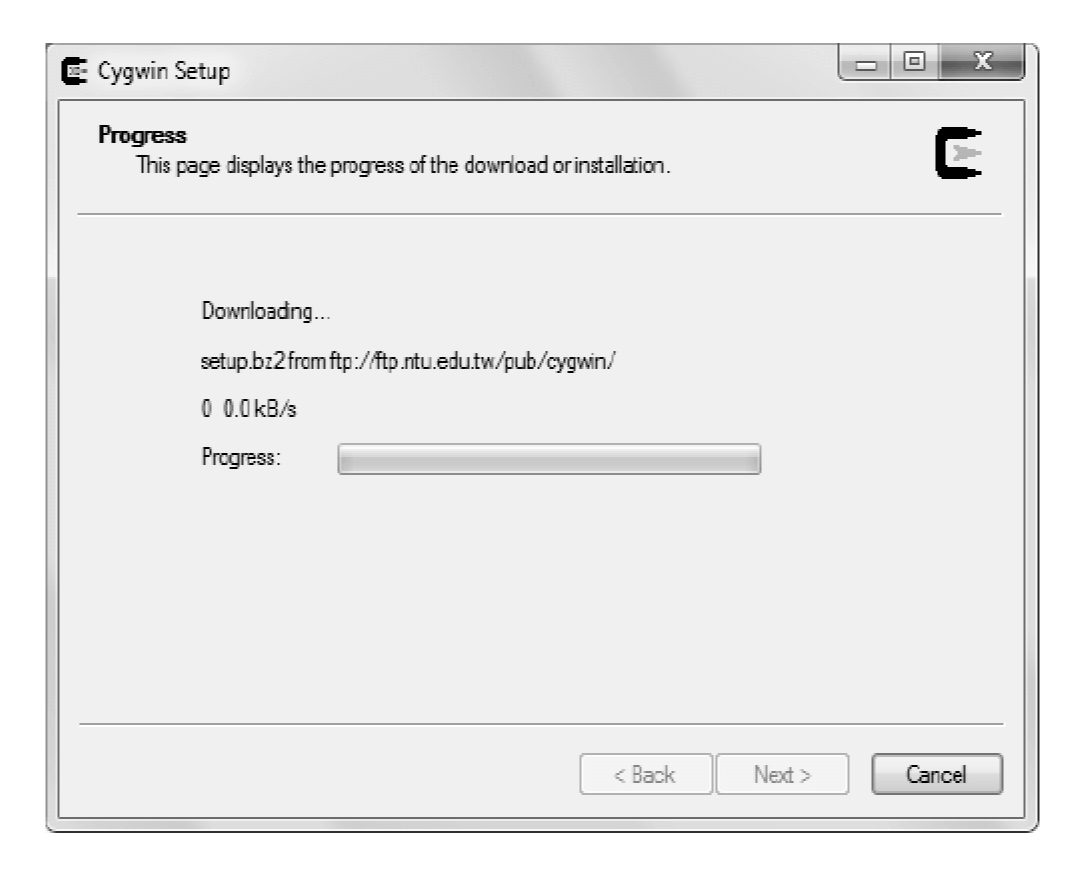


图 2-9　更新工具列表

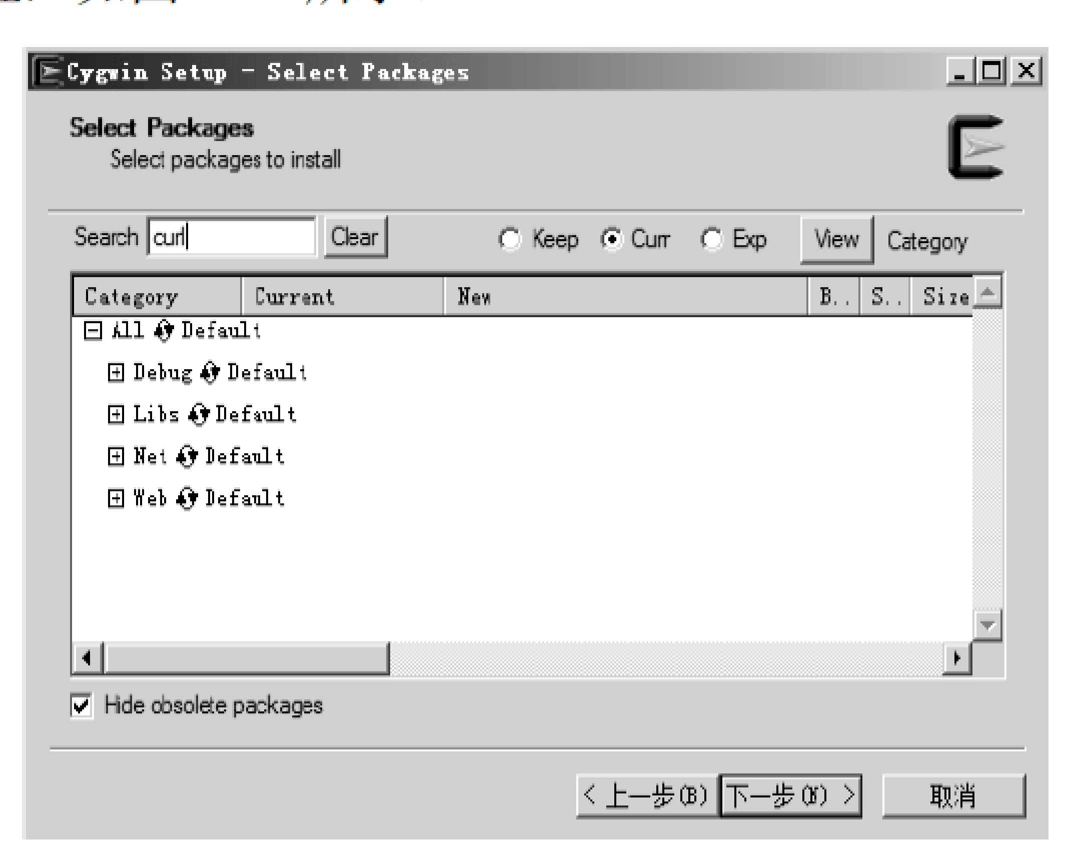


图 2-10　依次下载工具

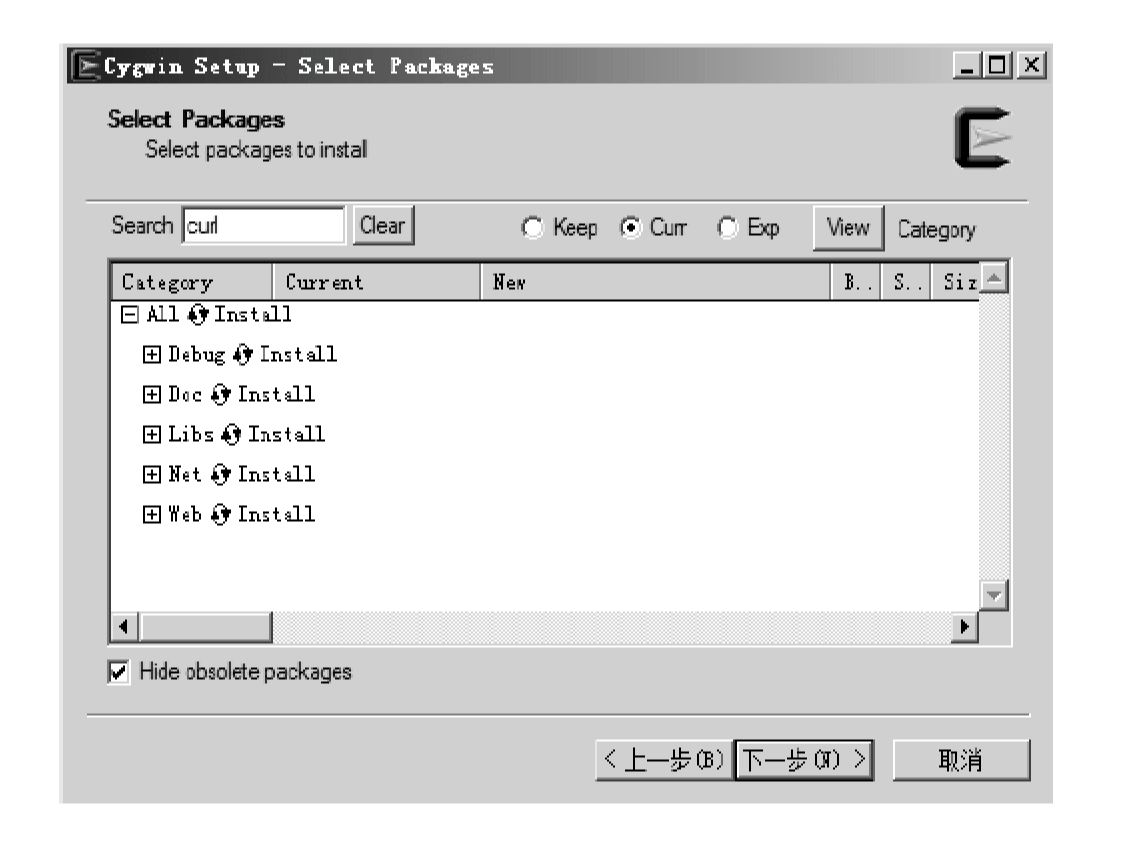


图 2-11　务必设置为 Install 形式

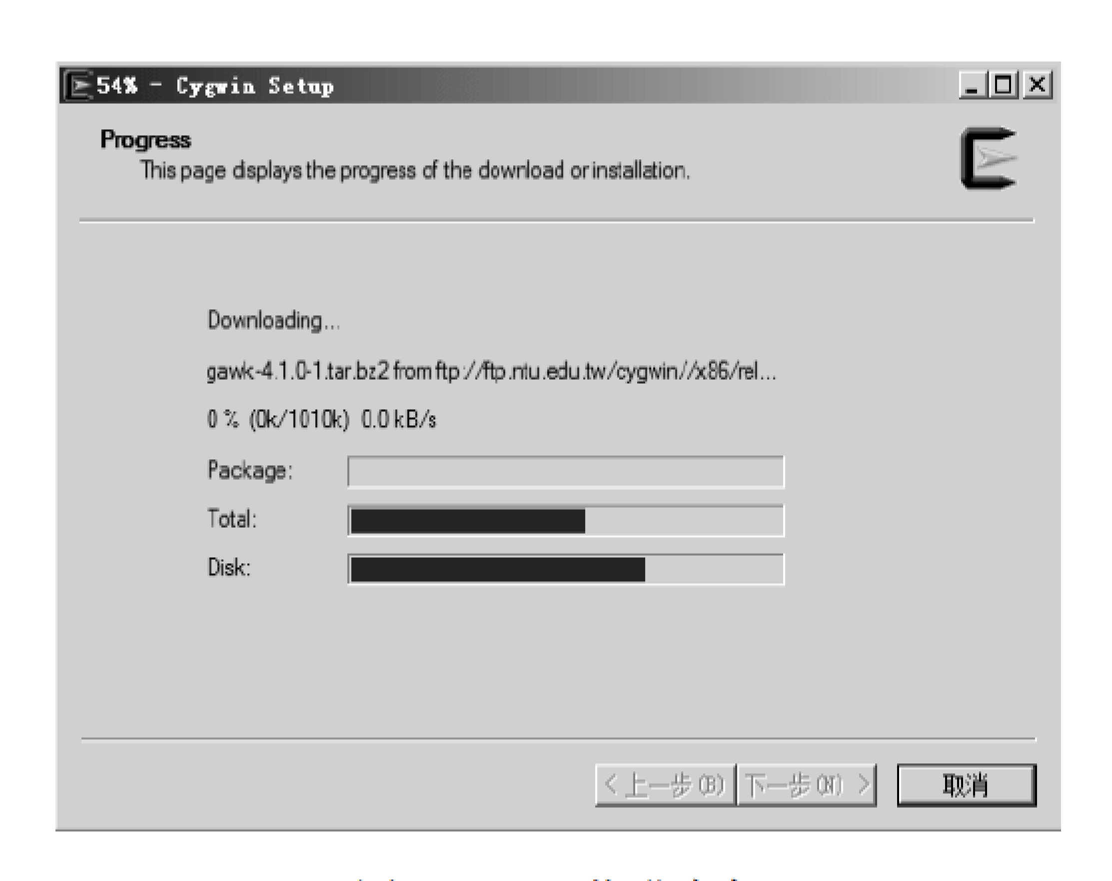


图 2-12　下载进度条

如果下载安装成功会出现提示信息，单击“完成”按钮即完成安装。当安装好 cygwin 后，打开 cygwin，会模拟出一个 Linux 的工作环境，然后按照 Linux 平台的源码下载方法就可以下载 Android 源码了。

建议读者在下载 Android 源码时，严格按照官方提供的步骤进行，地址是 http://source.android./source/downloading.html，这一点对初学者来说尤为重要。另外，整个下载过程比较漫长，需要大家耐心等待。例如，图 2-13 是笔者下载 Android 5.0 时的机器命令截图。

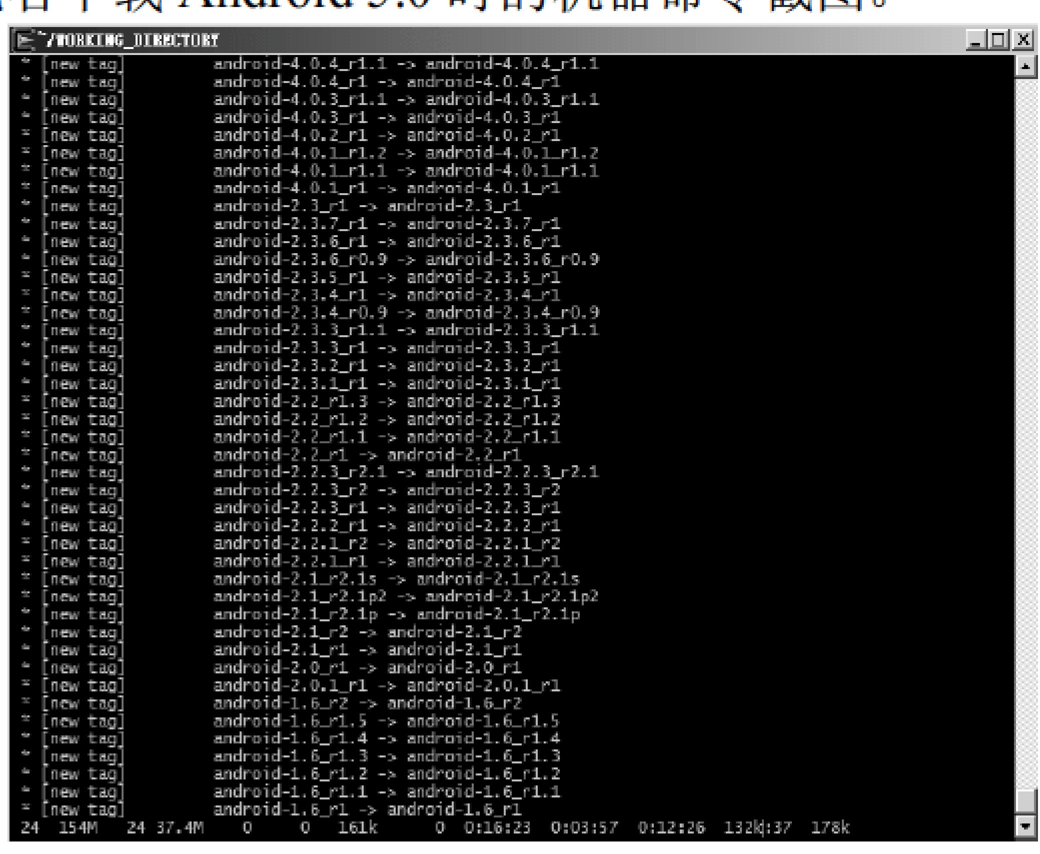


图 2-13　在 Windows 中用 cygwin 工具下载 Android 源码的截图

2.3 编译源码

知识点讲解：光盘:视频\知识点\第 2 章\编译源码.avi

编译 Android 源码的方法非常简单，只需使用 Android 源码根目录下的 Makefile，执行 make 命令即可轻松实现。当然在编译 Android 源码之前，首先要确定已经完成同步工作。进入 Android 源码目录使用 make 命令进行编译，使用此命令的格式如下：

```
$: cd ~/Android 5.0（这里的“Android 5.0”就是我们下载源码的保存目录）
$: make
```

编译 Android 源码可以得到“~/project/android/cupcake/out”目录，笔者的截图界面如图 2-14 所示。

```
注意: external/doclava/src/com/google/doclava/Stubs.java 使用了未经检查或不安全
的操作。
注意: 要了解详细信息，请使用 -Xlint:unchecked 重新编译。
Notice file: external/doclava/NOTICE -- out/host/linux-x86/obj/NOTICE_FILES/src/
/framework/doclava.jar.txt
Install: out/host/linux-x86/framework/doclava.jar
target Java: core (out/target/common/obj/JAVA_LIBRARIES/core_intermediates/class
es)
注意: 某些输入文件使用或覆盖了已过时的 API。
注意: 要了解详细信息，请使用 -Xlint:deprecation 重新编译。
注意: 某些输入文件使用了未经检查或不安全的操作。
注意: 要了解详细信息，请使用 -Xlint:unchecked 重新编译。
Copying: out/target/common/obj/JAVA_LIBRARIES/core_intermediates/classes-jarjar.
jar
Copying: out/target/common/obj/JAVA_LIBRARIES/core_intermediates/emma_out/lib/cl
asses-jarjar.jar
Copying: out/target/common/obj/JAVA_LIBRARIES/core_intermediates/classes.jar
target Java: conscrypt (out/target/common/obj/JAVA_LIBRARIES/conscrypt_intermedi
ates/classes)
注意: 某些输入文件使用或覆盖了已过时的 API。
注意: 要了解详细信息，请使用 -Xlint:deprecation 重新编译。
注意: 某些输入文件使用了未经检查或不安全的操作。
注意: 要了解详细信息，请使用 -Xlint:unchecked 重新编译。
host Prebuilt: jarjar (out/host/common/obj/JAVA_LIBRARIES/jarjar_intermediates/javalib.jar)
Install: out/host/linux-x86/framework/jarjar.jar
JarJar: out/target/common/obj/JAVA_LIBRARIES/conscrypt_intermediates/classes-jarjar.jar
Copying: out/target/common/obj/JAVA_LIBRARIES/conscrypt_intermediates/emma_out/lib/classes-jarjar.jar
Copying: out/target/common/obj/JAVA_LIBRARIES/conscrypt_intermediates/classes.jar
target Java: bouncycastle (out/target/common/obj/JAVA_LIBRARIES/bouncycastle_intermediates/classes)
注意: 某些输入文件使用或覆盖了已过时的 API。
注意: 要了解详细信息，请使用 -Xlint:deprecation 重新编译。
注意: 某些输入文件使用了未经检查或不安全的操作。
注意: 要了解详细信息，请使用 -Xlint:unchecked 重新编译。
JarJar: out/target/common/obj/JAVA_LIBRARIES/bouncycastle_intermediates/classes-jarjar.jar
Copying: out/target/common/obj/JAVA_LIBRARIES/bouncycastle_intermediates/emma_out/lib/classes-jarjar.jar
Copying: out/target/common/obj/JAVA_LIBRARIES/bouncycastle_intermediates/classes.jar
target Java: ext (out/target/common/obj/JAVA_LIBRARIES/ext_intermediates/classes)
```

图 2-14　编译过程的界面截图

整个编译过程也非常漫长，需要读者耐心等待。

2.3.1 搭建编译环境

在编译 Android 源码之前，需要先进行环境搭建工作。在接下来的内容中，以 Ubuntu 系统为例讲解搭建编译环境以及编译 Android 源码的方法。具体流程如下。

（1）安装 JDK，编译 Android 5.0 的源码需要 JDK 插件，可以下载 jdk-6u22-linux-i586.bin 后进行安装，对应命令如下：

```
$ cd /usr
$ mkdir java
$ cd java
$ sudo cp jdk-6u22-linux-i586.bin 所在目录 ./
$ sudo chmod 755 jdk-6u22-linux-i586.bin
$ sudo sh jdk-6u22-linux-i586.bin
```

（2）设置 JDK 环境变量，将如下环境变量添加到主文件夹目录下的.bashrc 文件中，然后用 source 命令使其生效，加入的环境变量代码如下：

```
export JAVA_HOME=/usr/java/jdk1.6.0_23
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=.:$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
export PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/bin/tools.jar:$JRE_HOME/bin
export ANDROID_JAVA_HOME=$JAVA_HOME
```

（3）安装需要的包，读者可以根据编译过程中的提示进行选择，可能需要的包的安装命令如下：

```
    $ sudo apt-get install git-core bison zlib1g-dev flex libx12-dev gperf sudo aptitude install git-core gnupg flex
bison gperf libsdl-dev libesd0-dev libwxgtk2.6-dev build-essential zip curl libncurses5-dev zlib1g-dev
```

2.3.2　开始编译

当所依赖的包安装完成之后，就可以开始编译 Android 源码了，具体步骤如下。

（1）首先进行编译初始化工作，在终端中执行下面的命令：

```
source build/envsetup.sh
```

或：

```
. build/envsetup.sh
```

执行后将会输出：

```
source build/envsetup.sh
including device/asus/grouper/vendorsetup.sh
including device/asus/tilapia/vendorsetup.sh
including device/generic/armv7-a-neon/vendorsetup.sh
including device/generic/armv7-a/vendorsetup.sh
including device/generic/mips/vendorsetup.sh
including device/generic/x86/vendorsetup.sh
including device/samsung/maguro/vendorsetup.sh
including device/samsung/manta/vendorsetup.sh
including device/samsung/toroplus/vendorsetup.sh
including device/samsung/toro/vendorsetup.sh
including device/ti/panda/vendorsetup.sh
including sdk/bash_completion/adb.bash
```

（2）然后选择编译目标，命令如下：

```
lunch full-eng
```

执行后会输出如下的提示信息。

```
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=5.0
TARGET_PRODUCT=full
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a
HOST_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-2.6.32-45-generic-x86_64-with-Ubuntu-10.04-lucid
HOST_BUILD_TYPE=release
BUILD_ID=JOP40C
OUT_DIR=out
============================================
```

（3）接下来开始编译代码，在终端中执行下面的命令：

```
make -j4
```

其中-j4 表示用 4 个线程进行编译。整个编译进度根据不同机器的配置而需要不同的时间。例如笔者电脑为 intel i5-2300 四核 2.8，4G 内存，经过近 4 小时才编译完成。当出现下面的信息时表示编译完成。

```
target Java: ContactsTests (out/target/common/obj/APPS/ContactsTests_intermediates/classes)
target Dex: Contacts
Done!
Install: out/target/product/generic/system/app/Browser.odex
Install: out/target/product/generic/system/app/Browser.apk
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Copying: out/target/common/obj/APPS/Contacts_intermediates/noproguard.classes.dex
target Package: Contacts (out/target/product/generic/obj/APPS/Contacts_intermediates/package.apk)
 'out/target/common/obj/APPS/Contacts_intermediates/classes.dex' as 'classes.dex'...
Processing target/product/generic/obj/APPS/Contacts_intermediates/package.apk
Done!
Install: out/target/product/generic/system/app/Contacts.odex
Install: out/target/product/generic/system/app/Contacts.apk
build/tools/generate-notice-files.py  out/target/product/generic/obj/NOTICE.txt  out/target/product/generic/obj/
NOTICE.html "Notices for files contained in the filesystem images in this directory:" out/target/product/generic/
obj/NOTICE_ FILES/src
Combining NOTICE files into HTML
Combining NOTICE files into text
Installed file list: out/target/product/generic/installed-files.txt
Target system fs image: out/target/product/generic/obj/PACKAGING/systemimage_intermediates/system.img
Running: mkyaffs2image -f out/target/product/generic/system out/target/product/generic/obj/PACKAGING/syste-
mimage_intermediates/system.img
Install system fs image: out/target/product/generic/system.img
DroidDoc took 5331 sec. to write docs to out/target/common/docs/doc-comment-check
```

2.3.3 在模拟器中运行

在模拟器中运行的步骤就比较简单了，只需在终端中执行下面的命令即可。

```
emulator
```

运行成功后的效果如图 2-15 所示。

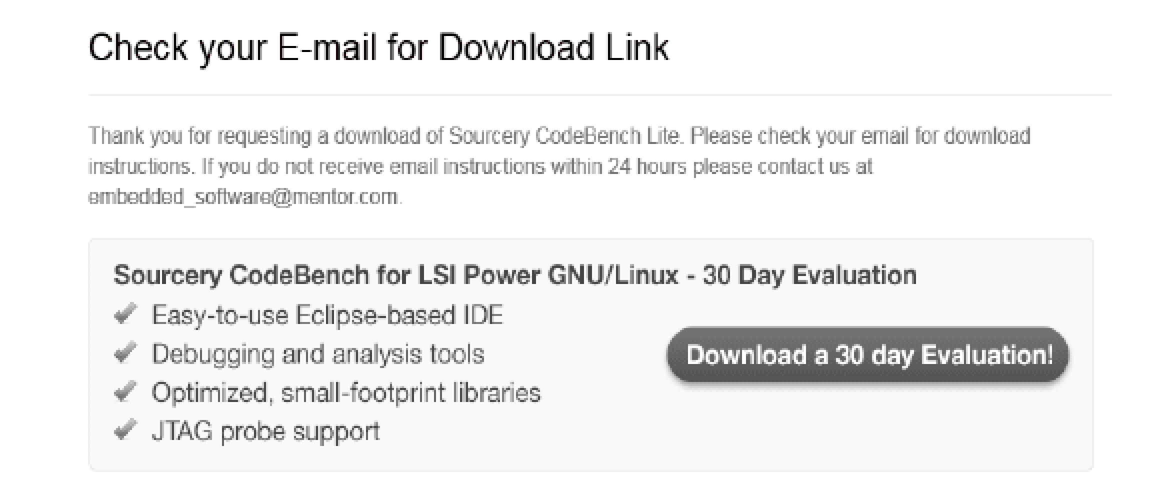
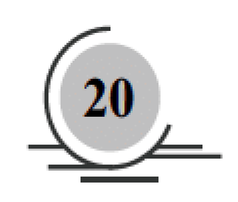


图 2-15 提示在邮件中得到下载地址

2.3.4 常见的错误分析

虽然编译方法非常简单，但是作为初学者来说很容易出错，在下面列出了其中常见的编译错误类型。



（1）缺少必要的软件

进入 Android 目录下，使用 make 命令编译，可能会发现出现如下错误提示。

```
host C: libneo_cgi <= external/clearsilver/cgi/cgi.c
external/clearsilver/cgi/cgi.c:22:18: error: zlib.h: No such file or directory
```

上述错误是因为缺少 zlib1g-dev，需要使用 apt-get 命令从软件仓库中安装 zlib1g-dev，具体命令如下：

```
sudo apt-get install zlib1g-dev
```

同理需要安装下面的软件，否则也会出现上述类似的错误。

```
sudo apt-get install flex
sudo apt-get install bison
sudo apt-get install gperf
sudo apt-get install libsdl-dev
sudo apt-get install libesd0-dev
sudo apt-get install libncurses5-dev
sudo apt-get install libx12-dev
```

（2）没有安装 Java 环境 JDK

当安装所有上述软件后，运行 make 命令再次编译 Android 源码。如果在之前忘记安装 Java 环境 JDK，则此时会出现很多 Java 文件无法编译的错误，如果打开 Android 的源码，可以看到在如下目录中下发现有很多 Java 源文件。

```
android/dalvik/libcore/dom/src/test/java/org/w3c/domts
```

这充分说明在编译 Android 之前必须先安装 Java 环境 JDK，安装流程如下。

- 从Oracle官方网站下载jdk-6u16-linux-i586.bin文件，然后安装。

在 Ubuntu 8.04 中，“/etc/profile”文件是全局的环境变量配置文件，它适用于所有的 shell。在登录 Linux 系统时应该先启动“/etc/profile”文件，然后再启动用户目录下的“~/.bash_profile”、“~/.bash_login”或“~/.profile”文件中的一个，执行的顺序和上面的排序一样。如果“~/.bash_profile”文件存在话，则还会执行“~/.bashrc”文件。在此只需要把 JDK 的目录放到“/etc/profile”目录下即可。

```
JAVA_HOME=/usr/local/src/jdk1.6.0_16
PATH=$PATH:$JAVA_HOME/bin:/usr/local/src/android-sdk-linux_x86-1.1_r1/tools:~/bin
```

- 重新启动机器，输入java –version命令，输出下面的信息则表示配置成功。

```
ava version "1.6.0_16"
Java(TM) SE Runtime Environment (build 1.6.0_16-b01)
Java HotSpot(TM) Client VM (build 13.2-b01, mixed mode, sharing)
```

当成功编译 Android 源码后，在终端会输出如下提示。

```
Target system fs image: out/target/product/generic/obj/PACKAGING/systemimage_unopt_intermediates/system.img
Install system fs image: out/target/product/generic/system.img
Target ram disk: out/target/product/generic/ramdisk.img
Target userdata fs image: out/target/product/generic/userdata.img
Installed file list: out/target/product/generic/installed-files.txt
root@dfsun2009-desktop:/bin/android#
```

2.4　实战演练——演示两种编译 Android 程序的方法

知识点讲解：光盘:视频\知识点\第 2 章\演示两种编译 Android 程序的方法.avi

Android 编译环境本身比较复杂，并且不像普通的编译环境那样只有顶层目录下才有 Makefile 文

件，而其他的每个 component 都使用统一标准的 Android.mk 文件。不过这并不是我们熟悉的 Makefile，而是经过 Android 自身编译系统的很多处理的。所以说要真正理清楚其中的联系还比较复杂，不过这种方式的好处在于，编写一个新的 Android.mk 给 Android 增加一个新的 Component 会变得比较简单。为了使读者更加深入地理解在 Linux 环境下编译 Android 程序的方法，在接下来的内容中，将分别演示两种编译 Android 程序的方法。

2.4.1 编译 Native C（本地 C 程序）的 helloworld 模块

编译 Java 程序可以直接采用 Eclipse 的集成环境来完成，实现方法非常简单，在这里就不再重复了。接下来将主要针对 C/C++进行说明，通过一个例子来讲解在 Android 中增加一个 C 程序的 Hello World 的方法。

（1）在$(YOUR_ANDROID)/development 目录下创建一个名为 hello 的目录，并用“$(YOUR_ANDROID)”指向 Android 源代码所在的目录。

```
- # mkdir $(YOUR_ANDROID)/development/hello
```

（2）在目录$(YOUR_ANDROID)/development/hello/下编写一个名为 hello.c 的 C 语言文件，文件 hello.c 的实现代码如下所示。

```
#include <stdio.h>
int main()
{
    printf("Hello World!\n");//输出 Hello World
return 0;
}
```

（3）在目录“$(YOUR_ANDROID)/development/hello/”下编写 Android.mk 文件。这是 Android Makefile 的标准命名，不能更改。文件 Android.mk 的格式和内容可以参考其他已有的 Android.mk 文件的写法，针对 helloworld 程序的 Android.mk 文件内容如下所示。

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)
LOCAL_SRC_FILES:= \
    hello.c
LOCAL_MODULE := helloworld
include $(BUILD_EXECUTABLE)
```

上述各个内容的具体说明如下。

- ❑ LOCAL_SRC_FILES：用来指定源文件用。
- ❑ LOCAL_MODULE：指定要编译的模块的名字，在下一步骤编译时将会用到。
- ❑ include $(BUILD_EXECUTABLE)：表示要编译成一个可执行文件，如果想编译成动态库则可用 BUILD_SHARED_LIBRARY，这些具体用法可以在$(YOUR_ANDROID)/build/core/config.mk 中查到。

（4）回到 Android 源代码顶层目录进行编译。

```
# cd $(YOUR_ANDROID) && make helloworld
```

在此需要注意，make helloworld 中的目标名 helloworld 就是上面 Android.mk 文件中由 LOCAL_MODULE 指定的模块名，最终的编译结果如下所示。

```
target thumb C: helloworld <= development/hello/hello.c
target  Executable:  helloworld  (out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/LINKED/
```



```
helloworld)
target Non-prelinked: helloworld (out/target/product/generic/symbols/system/bin/helloworld)
target Strip: helloworld (out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/helloworld)
Install: out/target/product/generic/system/bin/helloworld
```

（5）如果和上述编译结果相同，则编译后的可执行文件存放在如下目录：

```
out/target/product/generic/system/bin/helloworld
```

这样通过 adb push 将它传送到模拟器上，再通过 adb shell 登录到模拟器终端后就可以执行了。

2.4.2　手工编译 C 模块

前面讲解了通过标准的 Android.mk 文件来编译 C 模块的具体流程，其实也可以直接运用 gcc 命令行来编译 C 程序，这样可以更好地了解 Android 编译环境的细节。具体流程如下。

（1）在 Android 编译环境中，提供了 showcommands 选项来显示编译命令行，我们可以通过打开这个选项来查看一些编译时的细节。

（2）在具体操作之前需要使用如下命令把前面中的 helloworld 模块清除。

```
# make clean-helloworld
```

上面的“make clean-$(LOCAL_MODULE)”命令是 Android 编译环境提供的 make clean 的方式。

（3）使用 showcommands 选项重新编译 helloworld，具体命令如下所示。

```
  # make helloworld showcommands
  build/core/product_config.mk:229: WARNING: adding test OTA key
  target thumb C: helloworld <= development/hello/hello.c
prebuilt/linux-x86/toolchain/arm-eabi-5.0.1/bin/arm-eabi-gcc -I system/core/include -I hardware/libhardware/
include    -I hardware/ril/include    -I dalvik/libnativehelper/include    -I frameworks/base/include    -I external/
skia/include    -I out/target/product/generic/obj/include    -I bionic/libc/arch-arm/include    -I bionic/libc/include
-I bionic/libstdc++/include    -I bionic/libc/kernel/common    -I bionic/libc/kernel/arch-arm    -I bionic/libm/
include    -I bionic/libm/include/arch/arm    -I bionic/libthread_db/include    -I development/hello    -I
out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates    -c    -fno-exceptions -Wno-multichar
-march=armv5te -mtune=xscale -msoft-float -fpic -mthumb-interwork -ffunction-sections -funwind-tables
-fstack-protector -D__ARM_ARCH_5__ -D__ARM_ARCH_5T__ -D__ARM_ARCH_5E__ -D__ARM_ARCH_5TE__
-include system/core/include/arch/linux-arm/AndroidConfig.h -DANDROID -fmessage-length=0 -W -Wall
-Wno-unused -DSK_RELEASE -DNDEBUG -O2 -g -Wstrict-aliasing=2 -finline-functions
-fno-inline-functions-called-once -fgcse-after-reload -frerun-cse-after-loop -frename-registers -DNDEBUG
-UDEBUG -mthumb -Os -fomit-frame-pointer -fno-strict-aliasing -finline-limit=64        -MD -o
out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/hello.o development/hello/hello.c

target  Executable:  helloworld  (out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/LINKED/
helloworld)

prebuilt/linux-x86/toolchain/arm-eabi-5.0.1/bin/arm-eabi-g++ -nostdlib -Bdynamic -Wl,-T,build/core/armelf.x
-Wl,-dynamic-linker,/system/bin/linker -Wl,--gc-sections -Wl,-z,nocopyreloc -o out/target/product/generic/obj/
EXECUTABLES/helloworld_intermediates/LINKED/helloworld -Lout/target/product/generic/obj/lib -Wl,-rpath-link
=out/target/product/generic/obj/lib -lc -lstdc++ -lm    out/target/product/generic/obj/lib/crtbegin_ dynamic.o
out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/hello.o          -Wl,--no-undefined
prebuilt/linux-x86/toolchain/arm-eabi-5.0.1/bin/../lib/gcc/arm-eabi/5.0.1/interwork/libgcc.a
out/target/product/generic/obj/lib/crtend_android.o

target Non-prelinked: helloworld (out/target/product/generic/symbols/system/bin/helloworld)
```

```
out/host/linux-x86/bin/acp -fpt out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/LINKED/
helloworld out/target/product/generic/symbols/system/bin/helloworld

target Strip: helloworld (out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/helloworld)

  out/host/linux-x86/bin/soslim --strip --shady --quiet out/target/product/generic/symbols/system/bin/helloworld
--outfile out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/helloworld

Install: out/target/product/generic/system/bin/helloworld

  out/host/linux-x86/bin/acp -fpt out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/helloworld
out/target/product/generic/system/bin/helloworld
```

从上述命令行可以看到，Android 编译环境所用的交叉编译工具链如下所示。

```
prebuilt/linux-x86/toolchain/arm-eabi-5.0.1/bin/arm-eabi-gcc
```

其中参数-I 和-L 分别指定了所用的 C 库头文件和动态库文件路径分别是 bionic/libc/include 和 out/target/product/generic/obj/li，其他还包括很多编译选项以及-D 所定义的预编译宏。

（4）此时就可以利用上面的编译命令来手工编译 helloworld 程序，首先手工删除上次编译得到的 helloworld 程序。

```
# rm out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/hello.o
# rm out/target/product/generic/system/bin/helloworld
```

然后再用 gcc 编译以生成目标文件。

```
  # prebuilt/linux-x86/toolchain/arm-eabi-5.0.1/bin/arm-eabi-gcc -I bionic/libc/arch-arm/include -I bionic/libc/include -I
bionic/libc/kernel/common    -I bionic/libc/kernel/arch-arm -c   -fno-exceptions -Wno-multichar -march=armv5te
-mtune=xscale -msoft-float -fpic -mthumb-interwork -ffunction-sections -funwind-tables -fstack-protector -D__
ARM_ARCH_5__ -D__ARM_ARCH_5T__ -D__ARM_ARCH_5E__ -D__ARM_ARCH_5TE__ -include system/
core/include/arch/linux-arm/AndroidConfig.h -DANDROID -fmessage-length=0 -W -Wall -Wno-unused -DSK_
RELEASE -DNDEBUG -O2 -g -Wstrict-aliasing=2 -finline-functions -fno-inline-functions-called-once -fgcse-
after-reload -frerun-cse-after-loop -frename-registers -DNDEBUG -UDEBUG -mthumb -Os -fomit-frame- pointer
-fno-strict-aliasing -finline-limit=64        -MD -o out/target/product/generic/obj/EXECUTABLES/ helloworld_
intermediates/hello.o development/hello/hello.c
```

如果此时与 Android.mk 编译参数进行比较，会发现上面主要减少了不必要的参数-I。

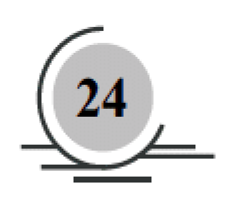
（5）接下来开始生成可执行文件。

```
  # prebuilt/linux-x86/toolchain/arm-eabi-5.0.1/bin/arm-eabi-gcc -nostdlib -Bdynamic -Wl,-T,build/core/armelf.x
-Wl,-dynamic-linker,/system/bin/linker -Wl,--gc-sections -Wl,-z,nocopyreloc -o out/target/product/generic/obj/
EXECUTABLES/helloworld_intermediates/LINKED/helloworld -Lout/target/product/generic/obj/lib
-Wl,-rpath-link=out/target/product/generic/obj/lib -lc -lm   out/target/product/generic/obj/EXECUTABLES/
helloworld_intermediates/hello.o out/target/product/generic/obj/lib/crtbegin_dynamic.
o -Wl,--no-undefined ./prebuilt/linux-x86/toolchain/arm-eabi-5.0.1/bin/../lib/gcc/arm-eabi/5.0.1/interwork/libgcc.
a out/target/product/generic/obj/lib/crtend_android.o
```

在此需要特别注意的是参数“-Wl,-dynamic-linker,/system/bin/linker”，它指定了 Android 专用的动态链接器是“/system/bin/linker”，而不是平常使用的 ld.so。

（6）最后可以使用命令 file 和 readelf 来查看生成的可执行程序。

```
  # file out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/LINKED/helloworld
  out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/LINKED/helloworld: ELF 32-bit LSB
executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), not stripped
  #   readelf  -d  out/target/product/generic/obj/EXECUTABLES/helloworld_intermediates/LINKED/helloworld
```



```
|grep NEEDED
 0x00000001 (NEEDED)                    Shared library: [libc.so]
 0x00000001 (NEEDED)                    Shared library: [libm.so]
```

这就是 ARM 格式的动态链接可执行文件，在运行时需要 libc.so 和 libm.so。当提示 not stripped 时表示它还没被 STRIP（剥离）。嵌入式系统中为节省空间通常将编译完成的可执行文件或动态库进行剥离，即去掉其中多余的符号表信息。在前面 make helloworld showcommands 命令的最后我们也可以看到，Android 编译环境中使用了 out/host/linux-x86/bin/soslim 工具进行 STRIP。

2.5　编译 Android Kernel

知识点讲解：光盘:视频\知识点\第 2 章\编译 Android Kernel.avi

编译 Android Kernel 代码就是编译 Android 内核代码，在进行具体编译工作之前需要先了解在 Android 开源系统中包含的如下 3 部分代码。

- 仿真器公共代码：对应的工程名是kernel/common.get。
- MSM平台的内核代码：对应的工程名是kernel/msm.get。
- OMAP平台的内核代码：对应的工程名是kernel/omap.get。

在本节的内容中，将详细讲解编译上述 Android Kernel 的基本知识。

2.5.1　获取 Goldfish 内核代码

Goldfish 是一种虚拟的 ARM 处理器，通常在 Android 的仿真环境中使用。在 Linux 的内核中，Goldfish 作为 ARM 体系结构的一种“机器”。在 Android 的发展过程中，Goldfish 内核的版本也从 Linux 2.6.25 升级到了 Linux 3.4，此处理器的 Linux 内核和标准的 Linux 内核有以下 3 个方面的差别。

- Goldfish机器的移植。
- Goldfish一些虚拟设备的驱动程序。
- Android中特有的驱动程序和组件。

Goldfish 处理器有两个版本，分别是 ARMv 5 和 ARMv 7，在一般情况下，只需使用 ARMv 5 版本即可。在 Android 开源工程的代码仓库中，使用 git 工具得到 Goldfish 内核代码的命令如下所示。

```
$ git clone git://android.git.kernel.org/kernel/common.git
```

在其 Linux 源代码的根目录中，配置和编译 Goldfish 内核的过程如下所示。

```
$make ARCH=arm goldfish_defconfig .config
$make ARCH=arm CROSS_COMPILE={path}/arm-none-linux-gnueabi-
```

其中，CROSS_COMPILE 的 path 值用于指定交叉编译工具的路径。

编译结果如下所示。

```
LD vmlinux
SYSMAP system.map
SYSMAP .tmp_system.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS arch/arm/boot/compressed/head.o
GZIP arch/arm/boot/compressed/piggy.gz
```

```
AS arch/arm/boot/compressed/piggy.o
CC arch/arm/boot/compressed/misc.o
LD arch/arm/boot/compressed/vmlinux
    OBJCONPY arch/arm/boot/zImage
    Kernel: arch/arm/boot/zImage is ready
```

- vmlinux：是Linux进行编译和连接之后生成的ELF格式的文件。
- Image：是未经过压缩的二进制文件。
- piggy：是一个解压缩程序。
- zImage：是解压缩程序和压缩内核的组合。

在 Android 源代码的根目录中，vmlinux 和 zImage 分别对应 Android 代码 prebuilt 中预编译的 ARM 内核。使用 zImage 可以替换 prebuilt 中“prebuilt/android-arm/”目录下的 goldfish_defconfig，此文件的主要片断如下所示。

```
CONFIG_ARM=y
#
# System Type
#
CONFIG_ARCH_GOLDFISH=y
#
# Goldfish options
#
CONFIG_MACH_GOLDFISH=y
# CONFIG_MACH_GOLDFISH_ARMV7 is not set
```

因为 Goldfish 是 ARM 处理器，所以 CONFIG_ARM 宏需要被使能，CONFIG_ARCH_GOLDFISH 和 CONFIG_MACH_GOLDFISH 宏是 Goldfish 处理器这类机器使用的配置宏。

在 goldfish_defconfig 中，与 Android 系统相关的宏如下所示。

```
#
# android
#
CONFIG_ANDROID=y
CONFIG_ANDROID_BUNDER_IPC=y           #binder ipc 驱动程序
CONFIG_ANDROID_LOGGER=y               #log 记录器驱动程序
# CONFIG_ANDROID_RAM_CONSOLE is not set
CONFIG_ANDROID_TIMED_OUTPUT=y         #定时输出驱动程序框架
CONFIG_ANDROID_LOW_MEMORY_KILLER=y
CONFIG_ANDROID_PMEM=y                 #物理内存驱动程序
CONFIG_ASHMEM=y                       #匿名共享内存驱动程序
CONFIG_RTC_INTF_ALARM=y
CONFIG_HAS_WAKELOCK=y 电源管理相关的部分 wakelock 和 earlysuspend
CONFIG_HAS_EARLYSUSPEND=y
CONFIG_WAKELOCK=y
CONFIG_WAKELOCK_STAT=y
CONFIG_USER_WAKELOCK=y
CONFIG_EARLYSUSPEND=y
```

goldfish_defconfig 配置文件中，另外有一个宏是处理器虚拟设备的“驱动程序”，其内容如下所示。

```
CONFIG_MTD_GOLDFISH_NAND=y
CONFIG_KEYBOARD_GOLDFISH_EVENTS=y
CONFIG_GOLDFISH_TTY=y
```

```
CONFIG_BATTERY_GOLDFISH=y
CONFIG_FB_GOLDFISH=y
CONFIG_MMC_GOLDFISH=y
CONFIG_RTC_DRV_GOLDFISH=y
```

在 Goldfish 处理器的各个配置选项中，体系结构和 Goldfish 的虚拟驱动程序是基于标准 Linux 的内容的驱动程序框架，但是这些设备在不同的硬件平台的移植方式不同；Android 专用的驱动程序是 Android 中特有的内容，非 Linux 标准，但是和硬件平台无关。

和原 Linux 内核相比，Android 内核增加了 Android 的相关 Driver，对应的目录如下所示。

```
kernel/drivers/android
```

主要分为以下几类 Driver。

- Android IPC系统：Binder (binder.c)。
- Android 日志系统：Logger (logger.c)。
- Android 电源管理：Power (power.c)。
- Android 闹钟管理：Alarm (alarm.c)。
- Android 内存控制台：Ram_console (ram_console.c)。
- Android 时钟控制的gpio：Timed_gpio (timed_gpio.c)。

对于本书讲解的驱动程序开发来说，我们比较关心的是 Goldfish 平台下相关的驱动文件，具体说明如下。

（1）字符输出设备：

```
kernel/drivers/char/goldfish_tty.c
```

（2）图像显示设备（Frame Buffer）：

```
kernel/drivers/video/goldfishfb.c
```

（3）键盘输入设备文件：

```
kernel/drivers/input/keyboard/goldfish_events.c
```

（4）RTC（Real Time Clock）设备文件：

```
kernel/drivers/rtc/rtc-goldfish.c
```

（5）USB Device 设备文件：

```
kernel/drivers/usb/gadget/android_adb.c
```

（6）SD 卡设备文件：

```
kernel/drivers/mmc/host/goldfish.c
```

（7）FLASH 设备文件：

```
kernel/drivers/mtd/devices/goldfish_nand.c
kernel/drivers/mtd/devices/goldfish_nand_reg.h
```

（8）LED 设备文件：

```
kernel/drivers/leds/ledtrig-sleep.c
```

（9）电源设备：

```
kernel/drivers/power/goldfish_battery.c
```

（10）音频设备：

```
kernel/arch/arm/mach-goldfish/audio.c
```

（11）电源管理：

```
kernel/arch/arm/mach-goldfish/pm.c
```

（12）时钟管理：

```
kernel/arch/arm/mach-goldfish/timer.c
```

2.5.2 获取 MSM 内核代码

在目前市面中，谷歌的手机产品 G1 是基于 MSM 内核的，MSM 是高通公司的应用处理器，在 Android 代码库中公开了对应的 MSM 的源代码。在 Android 开源工程的代码仓库中，使用 git 工具得到 MSM 内核代码的命令如下所示。

```
$ git clone git://android.git.kernel.org/kernel/msm.git
```

2.5.3 获取 OMAP 内核代码

OMAP 是德州仪器公司的应用处理器，为 Android 使用的是 OMAP3 系列的处理器。在 Android 代码库中公开了对应的 MSM 的源代码，使用 git 工具得到 MSM 内核代码的命令如下所示。

```
$ git clone git://android.git.kernel.org/kernel/omap.git
```

2.5.4 编译 Android 的 Linux 内核

了解了上述 3 类 Android 内核后，下面开始讲解编译 Android 内核的方法。在此以 Ubuntu 8.10 为例，完整编译 Android 内核的流程如下所示。

（1）构建交叉编译环境

Android 的默认硬件处理器是 ARM，因此我们需要在自己的机器上构建交叉编译环境。交叉编译器 GNU Toolchain for ARM Processors 下载地址如下所示。

```
http://www.codesourcery.com/gnu_toolchains/arm/download.html
```

单击 GNU/Linux 对应的链接，输入验证邮箱后可以获得下载地址，如图 2-16 所示。

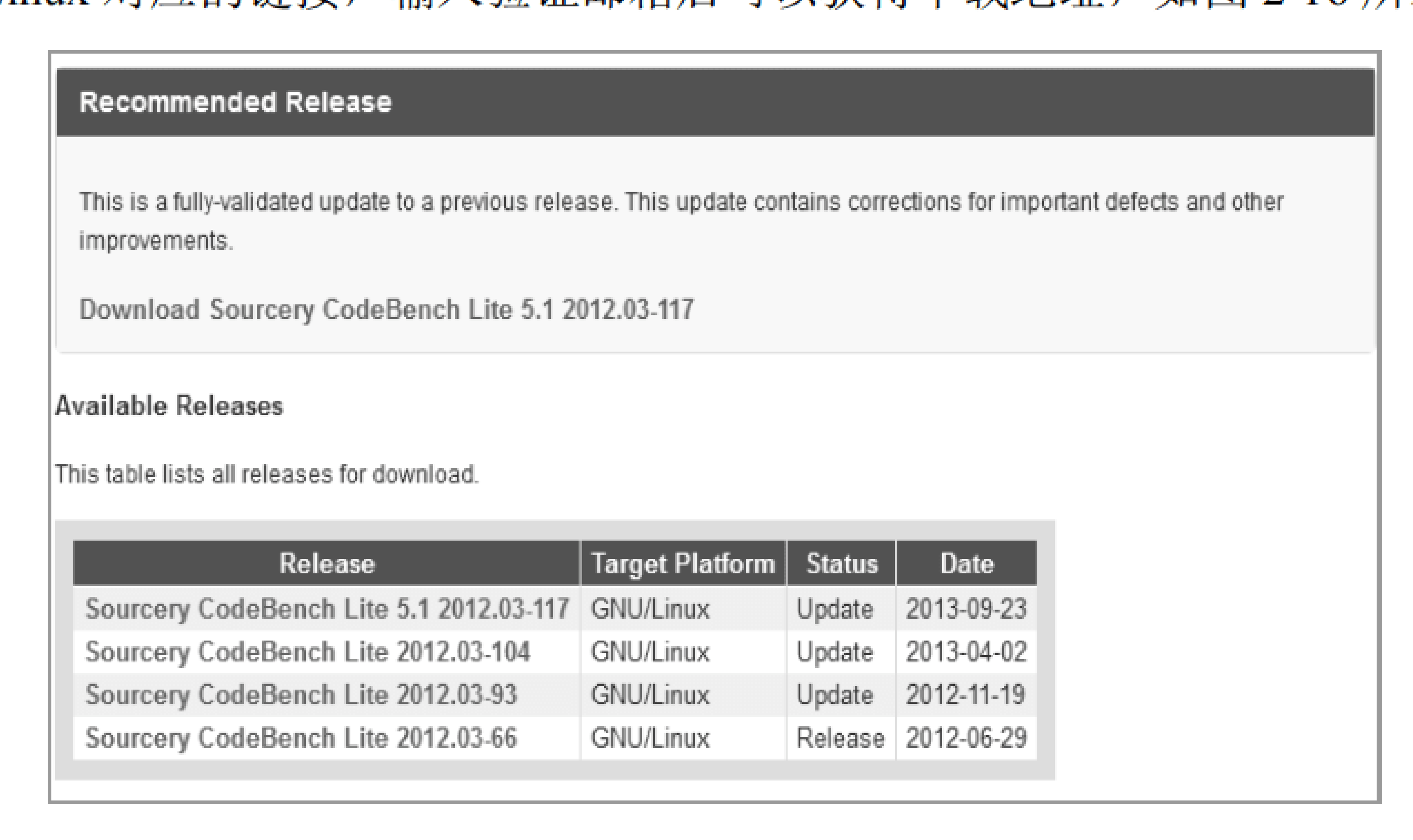


图 2-16 提示在邮件中得到下载地址

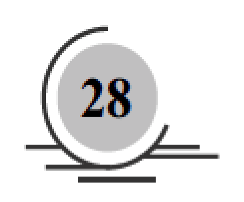
把下载到的“gnu.tar.bz2”文件解压到一目录下，例如“~/programes/”，并加入 PATH 环境变量：

```
vim ~/.bashrc
```

然后添加：

```
ARM_TOOLCHIAN=~/programes/arm-2008q3/bin/
export PATH=${PATH}:${ARM_TOOLCHIAN};
```

保存后并执行“soure~/.bashrc”命令。



（2）获取内核源码

源码地址如下所示。

```
http://code.google.com/p/android/downloads/list
```

选择的内核版本要与选用的模拟器版本尽量一致。下载后并解压后得到 kernel.git 文件夹：

```
tar -xvf ~/download/linux-3.2.5-android-4.3_r1.tar.gz
```

（3）获取内核编译配置信息文件

编译内核时需要使用 configure，通常 configure 有很多选项，我们往往不知道需要哪些选项。在运行 Android 模拟器时，有一个文件“/proc/config.gz”，这是当前内核的配置信息文件，把 config.gz 获取并解压放到“kernel.git/”下，然后改名为.config。命令如下所示。

```
cd kernel.git/
emulator &
adb pull /proc/config.gz
gunzip config.gz
mv config .config
```

（4）修改 Makefile

修改 195 行的代码：

```
CROSS_COMPILE        = arm-none-linux-gnueabi-
```

将 CROSS_COMPILE 值改为 arm-none-linux-gnueabi-，这是我们安装的交叉编译工具链的前缀，修改此处意在告诉 make 在编译时要使用该工具链。然后注释掉 562 和 563 行的如下代码：

```
#LDFLAGS_BUILD_ID = $(patsubst -Wl$(comma)%,%,/
#                          $(call ld-option, -Wl$(comma)--build-id,))
```

必须将上述代码中的 build id 值注释掉，因为目前版本的 Android 内核不支持该选项。

（5）编译

使用 make 进行编译，并同时生成 zImage：

```
LD         arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
```

这样生成 zImage 大小为 1.23M，android- sdk-linux_x86-4.3_r1/tools/lib/images/kernel-qemu 是 1.24M。

（6）使用模拟器加载内核测试

命令如下所示：

```
cd android/out/cupcake/out/target/product/generic
emulator -image system.img -data userdata.img -ramdisk ramdisk.img -kernel ~/project/android/kernel.git/
arch/arm/boot/zImage &
```

到此为止，模拟器就加载成功了。

第 3 章　搭建 Android 应用开发环境

应用层位于 Android 体系的最顶层，通常使用 Java 语言进行开发。平常说的 Android 应用开发是指开发能在 Android 系统上运行的程序，例如在手机中经常使用的斗地主游戏、谷歌地图等程序都属于应用开发层。本章将详细讲解搭建 Android 应用开发环境的知识，为读者学习后面知识打下基础。

3.1　搭建前的准备

知识点讲解：光盘:视频\知识点\第 3 章\搭建前的准备.avi

Android SDK 是开发 Android 应用程序所必须具备的工具，在搭建之前需要先确定基于 Android 应用软件所需要开发环境的要求，具体如表 3-1 所示。

表 3-1　开发系统所需求的参数

项　　目	版 本 要 求	说　　明	备　　注
操作系统	Windows XP 以上或 Vista Mac OS X 10.4.8+Linux Ubuntu Drapper 以上	根据自己的电脑自行选择	选择自己最熟悉的操作系统
软件开发包	Android SDK	选择最新版本的 SDK	截止到目前，最新版本是 5.0
IDE	Eclipse IDE+ADT	Eclipse3.3（Europa），3.4（Ganymede）ADT（Android Development Tools）开发插件	选择 for Java Developer
其他	JDK Apache Ant	Java SE Development Kit 5 或 6 Linux 和 Mac 上使用 Apache Ant 1.6.5+, Windows 上使用 1.7+版本	单独的 JRE 不可以的，必须要有 JDK，不兼容 Gnu Java 编译器（gcj）

Android 工具是由多个开发包组成的，具体说明如下：

- JDK：可以到网址http://java.sun.com/javase/downloads/index.jsp下载。
- Eclipse（Europa）：可以到网址http://www.eclipse.org/downloads/下载Eclipse IDE for Java Developers。
- Android SDK：可以到网址http://developer.android.com下载。
- 还有对应的开发插件。

3.2　安装 JDK

知识点讲解：光盘:视频\知识点\第 3 章\安装 JDK.avi

JDK（Java Development Kit）是整个 Java 的核心，包括了 Java 运行环境、Java 工具和 Java 基础的

类库。JDK 是学好 Java 的第一步，是开发和运行 Java 环境的基础，当用户要对 Java 程序进行编译的时候，必须先获得对应操作系统的 JDK，否则将无法编译 Java 程序。在安装 JDK 之前需要先获得 JDK，获得 JDK 的操作流程如下。

（1）登录 Oracle 官方网站，网址为 http://developers.sun.com/downloads/，如图 3-1 所示。

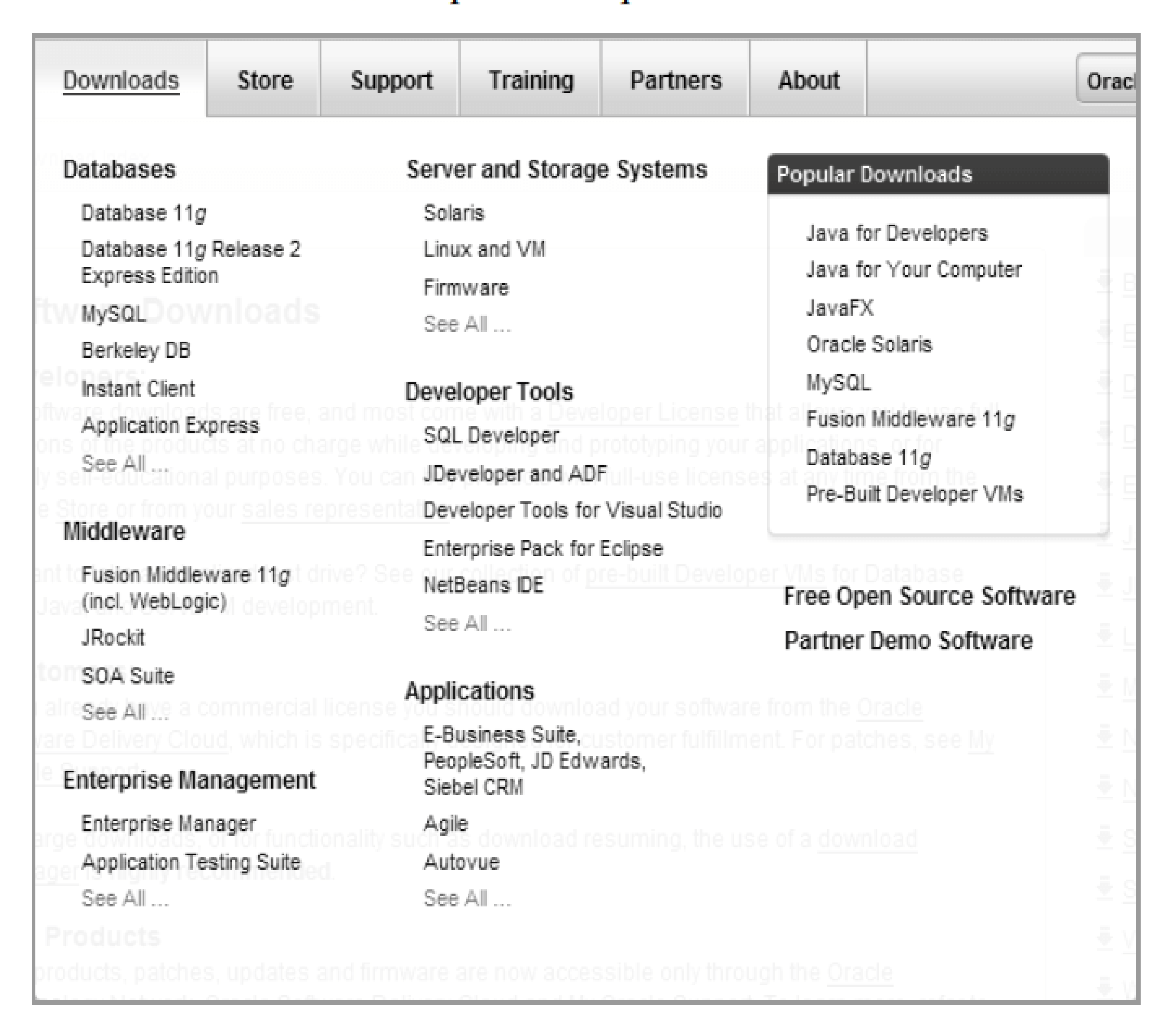


图 3-1　Oracle 官方下载页面

（2）在图 3-1 中可以看到有很多版本，在此选择当前最新的版本 Java 7，下载页面如图 3-2 所示。

（3）在图 3-2 中单击 JDK 下方的 Download 按钮，在弹出的新界面中选择将要下载的 JDK，笔者在此选择的是 Windows X86 版本，如图 3-3 所示。

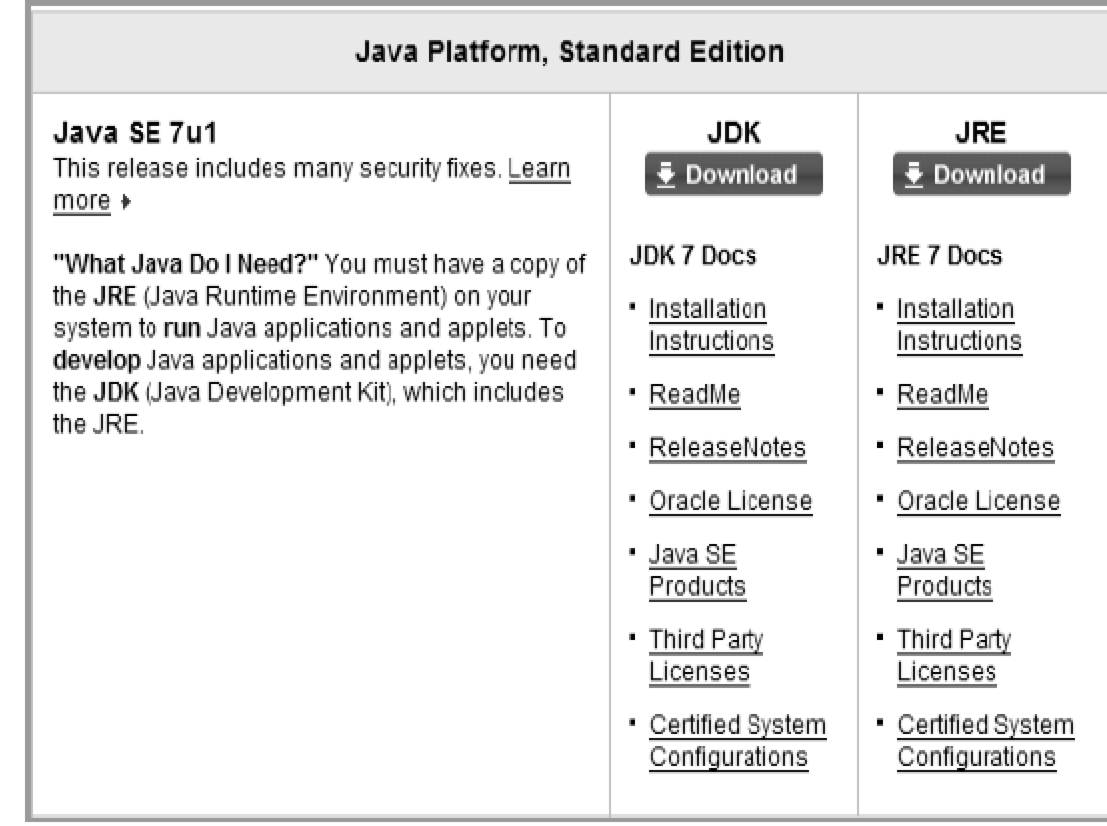


图 3-2　JDK 下载页面

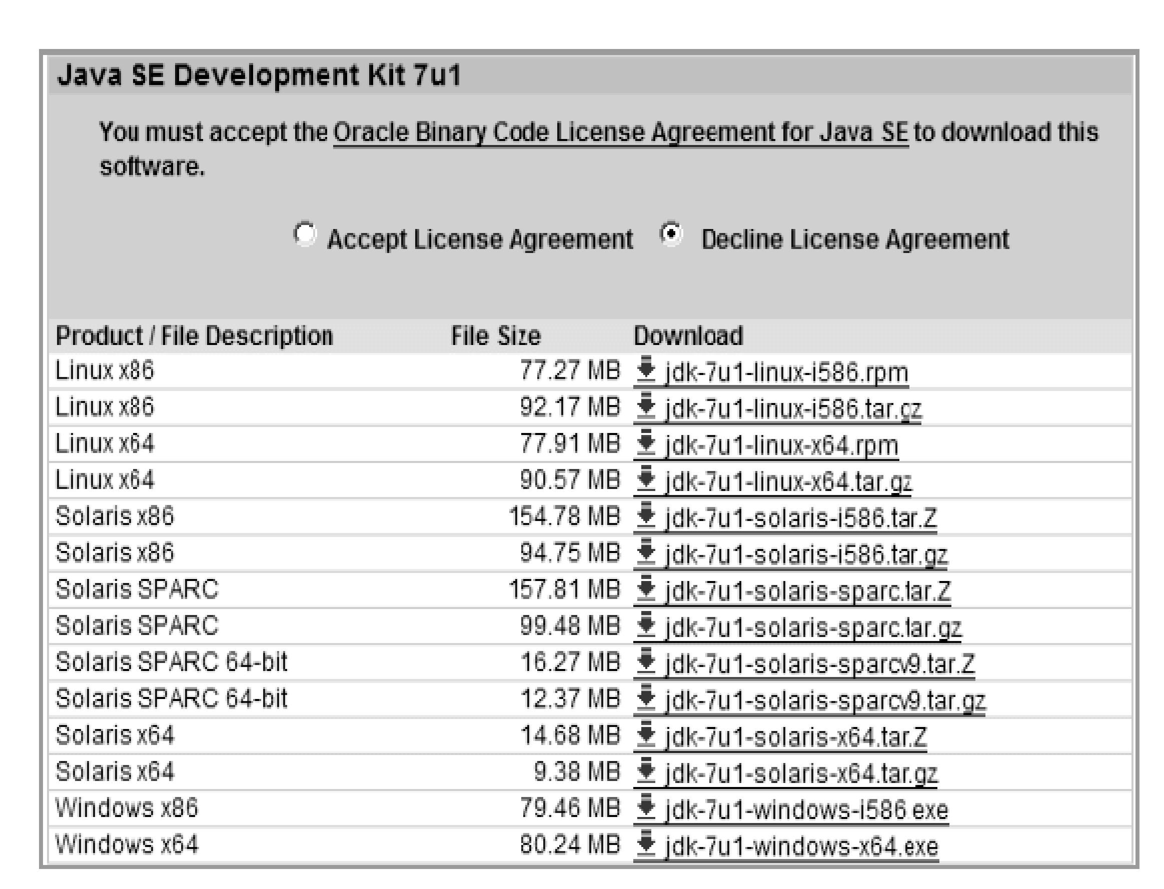


图 3-3　选择 Windows X86 版本

（4）下载完成后双击下载的“.exe”文件开始进行安装，将弹出安装向导对话框，在此单击“下一步”按钮，如图 3-4 所示。

（5）弹出自定义安装路径对话框，在此可以自定义选择，如图 3-5 所示。



图 3-4 “许可证协议”对话框

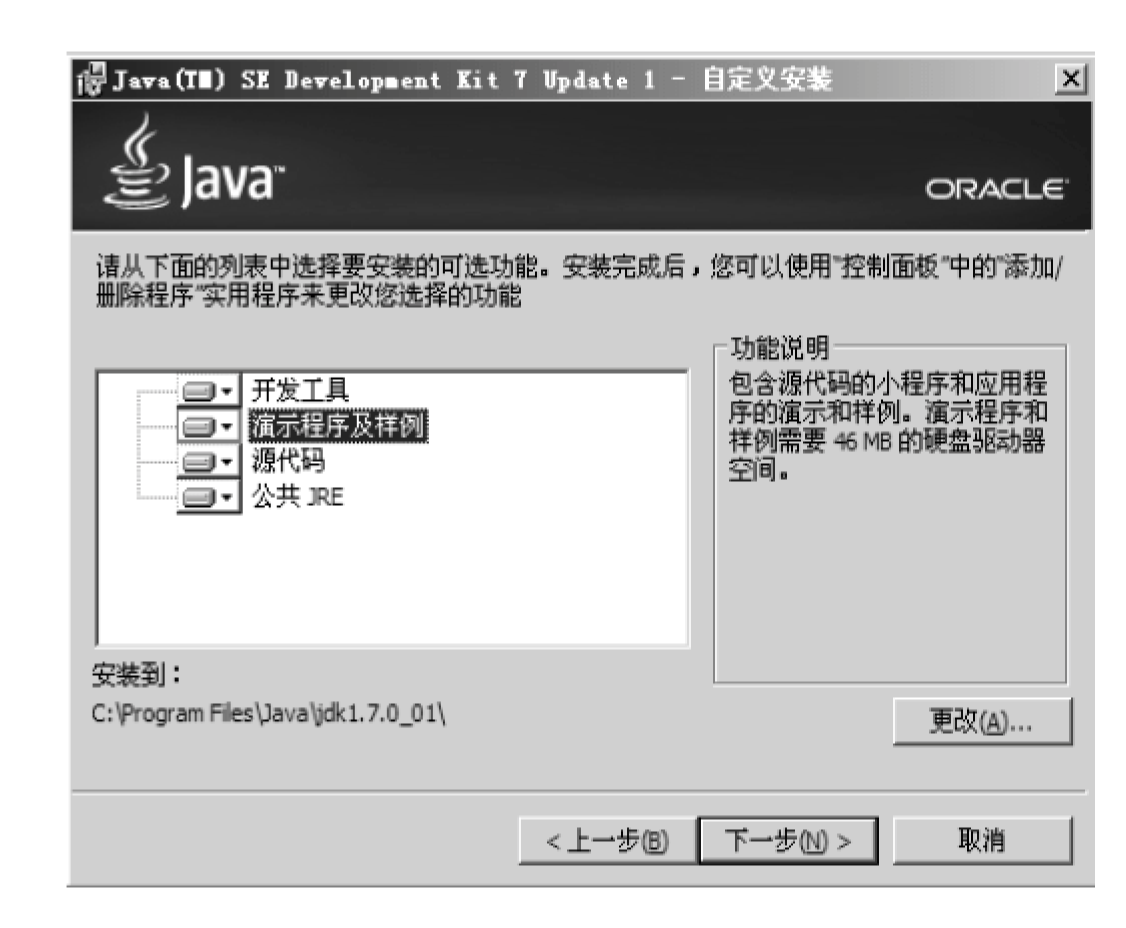


图 3-5 “安装路径”对话框

（6）单击“下一步”按钮后开始解压缩下载的文件，如图 3-6 所示。

（7）完成后弹出“目标文件夹”对话框，在此选择要安装的位置，如图 3-7 所示。

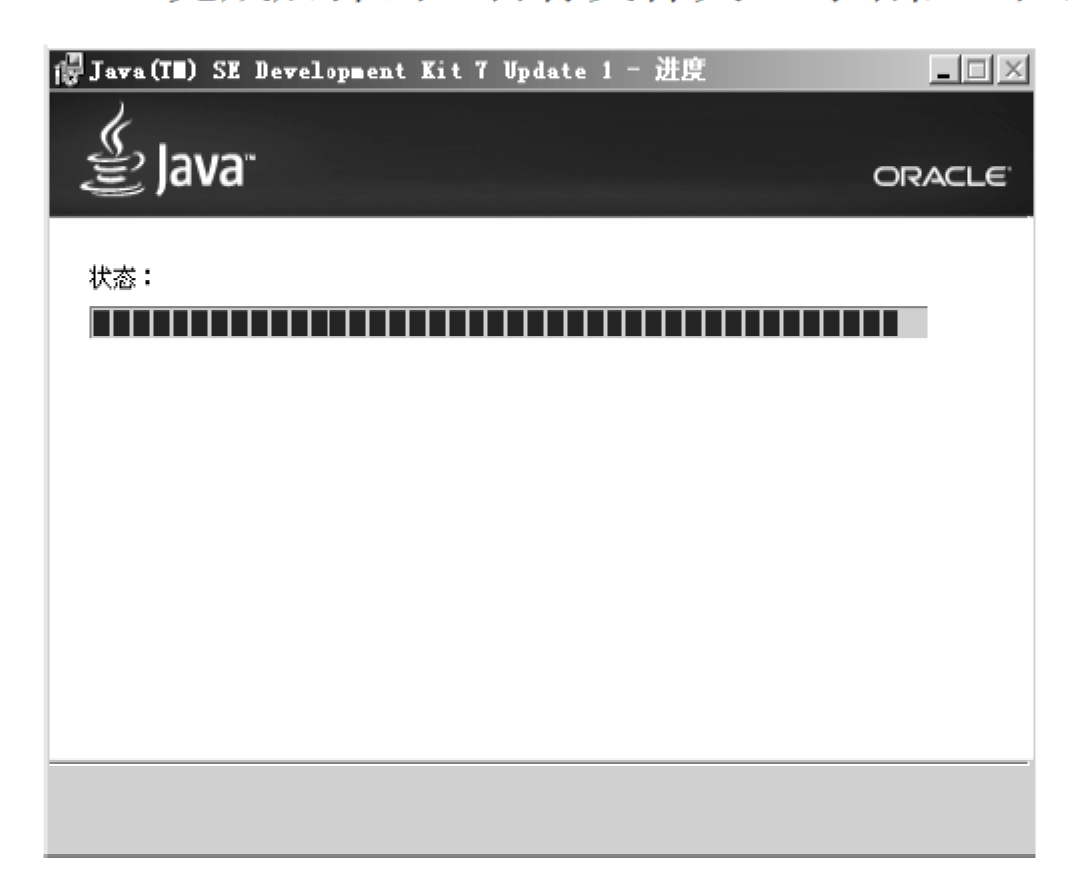


图 3-6 解压缩下载的文件



图 3-7 “目标文件夹”对话框

（8）单击“下一步”按钮后开始正式安装，如图 3-8 所示。

（9）完成后弹出“完成”对话框，单击“完成”按钮后完成整个安装过程，如图 3-9 所示。

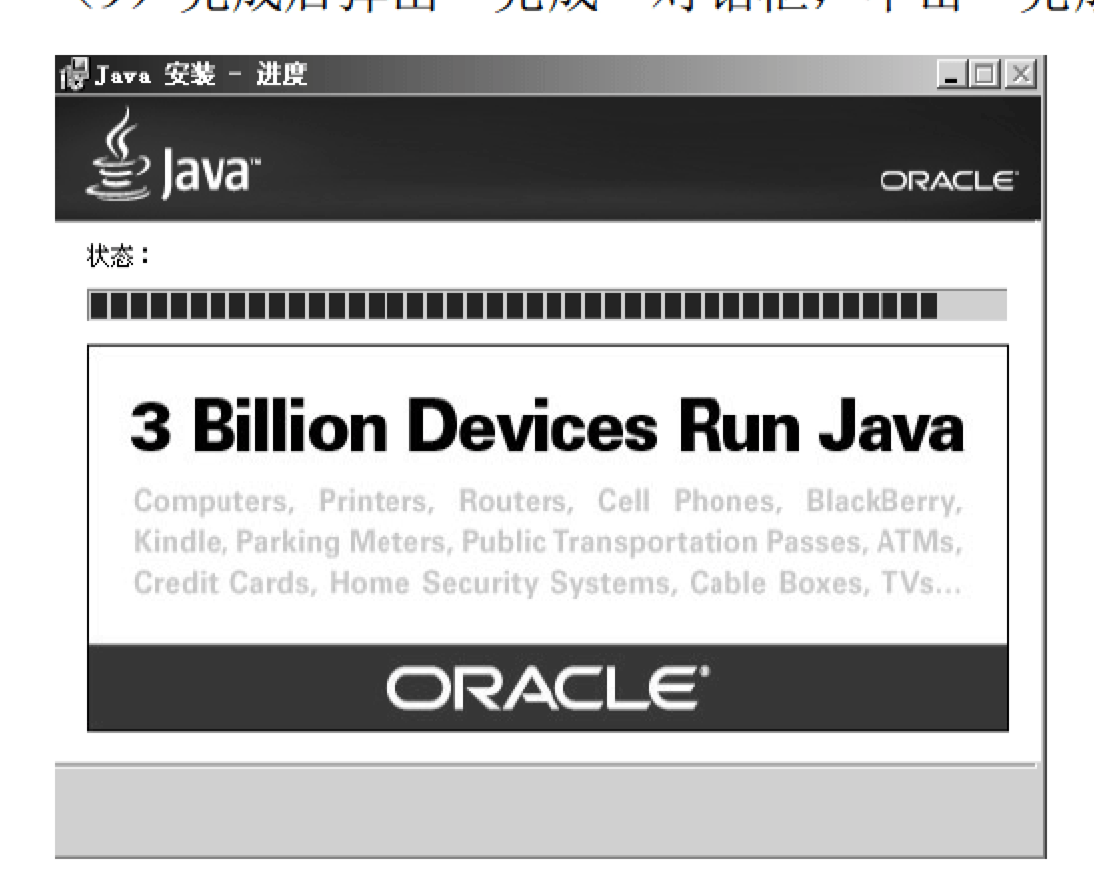
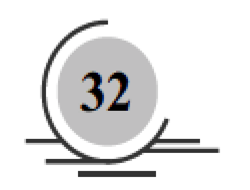


图 3-8 继续安装



图 3-9 完成安装



完成安装后可以检测是否安装成功，检测方法是选择“开始”｜“运行”命令，在运行框中输入 cmd 并回车，在打开的 CMD 窗口中输入 java –version，如果显示如图 3-10 所示的提示信息，则说明安装成功。

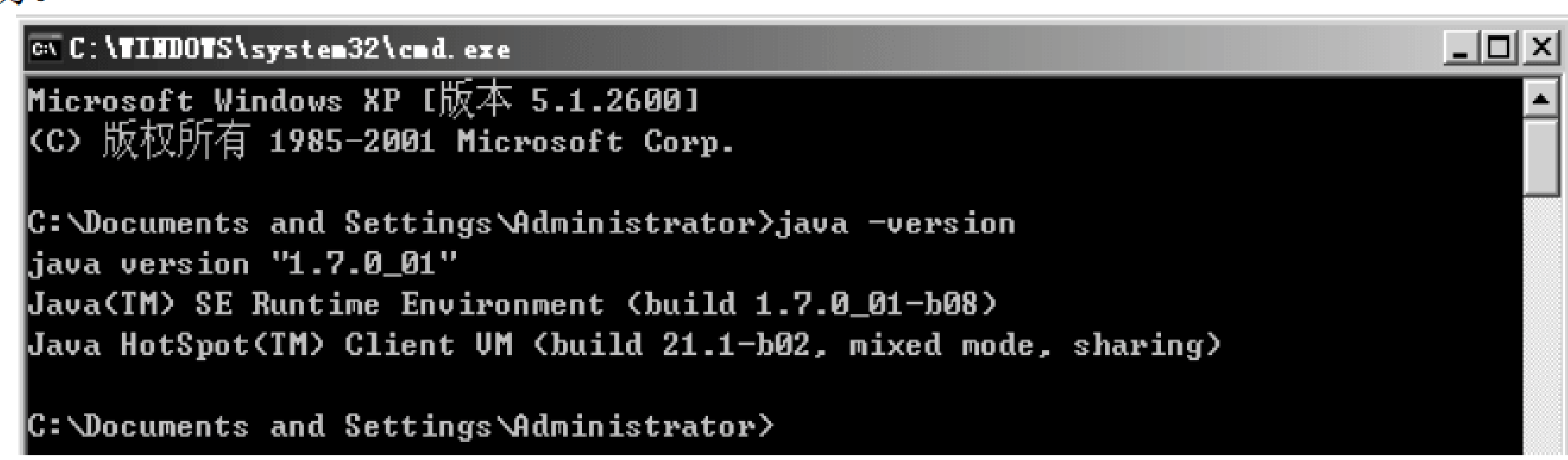


图 3-10　CMD 窗口

如果检测没有安装成功，需要将其目录的绝对路径添加到系统的 PATH 中。具体做法如下。

（1）右键单击“我的电脑”，在弹出的快捷菜单中选择“属性”｜“高级”命令，单击下面的“环境变量”按钮，在下面的“系统变量”处选择新建，在弹出对话框的“变量名”处输入 JAVA_HOME，“变量值”中输入刚才的目录，比如设置为\Java\jdk1.7.0_01，如图 3-11 所示。

（2）再次新建一个变量名为 classpath，其变量值如下所示：

```
.;%JAVA_HOME%/lib/rt.jar;%JAVA_HOME%/lib/tools.jar
```

单击“确定”按钮找到 PATH 的变量，双击或单击编辑，在变量值最前面添加如下值：

```
%JAVA_HOME%/bin;
```

具体如图 3-12 所示。

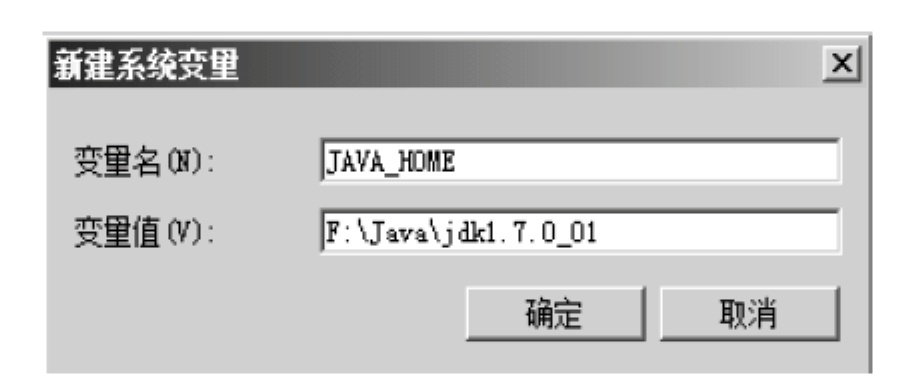


图 3-11　设置系统变量

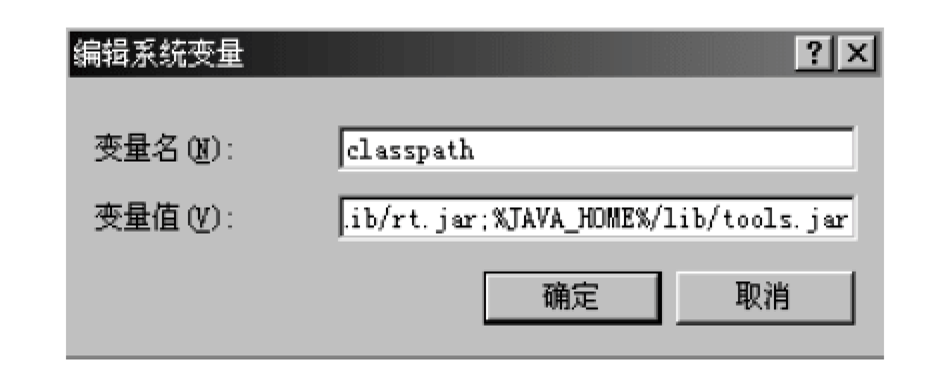


图 3-12　设置系统变量

（3）再选择“开始”｜“运行”命令，在运行框中输入 cmd 并回车，在打开的 CMD 窗口中输入 java –version，如果显示如图 3-13 所示的提示信息，则说明安装成功。



图 3-13　CMD 窗口

注意： 上述变量设置中，是按照笔者的安装路径设置的，笔者安装的 JDK 的路径是 C:\Program Files\Java\jdk1.7.0_02。

3.3 获取并安装 Eclipse 和 Android SDK

知识点讲解：光盘:视频\知识点\第 3 章\获取并安装 Eclipse 和 Android SDK.avi

在安装好 JDK 后，接下来需要安装 Eclipse 和 Android SDK。Eclipse 是进行 Android 应用开发的一个集成工具，而 Android SDK 是开发 Android 应用程序必须具备的框架。在 Android 官方公布的最新版本中，已经将 Eclipse 和 Android SDK 这两个工具进行了集成，一次下载即可同时获得这两个工具。

获取并安装 Eclipse 和 Android SDK 的具体步骤如下。

（1）登录 Android 的官方网站 http://developer.android.com/index.html，如图 3-14 所示。

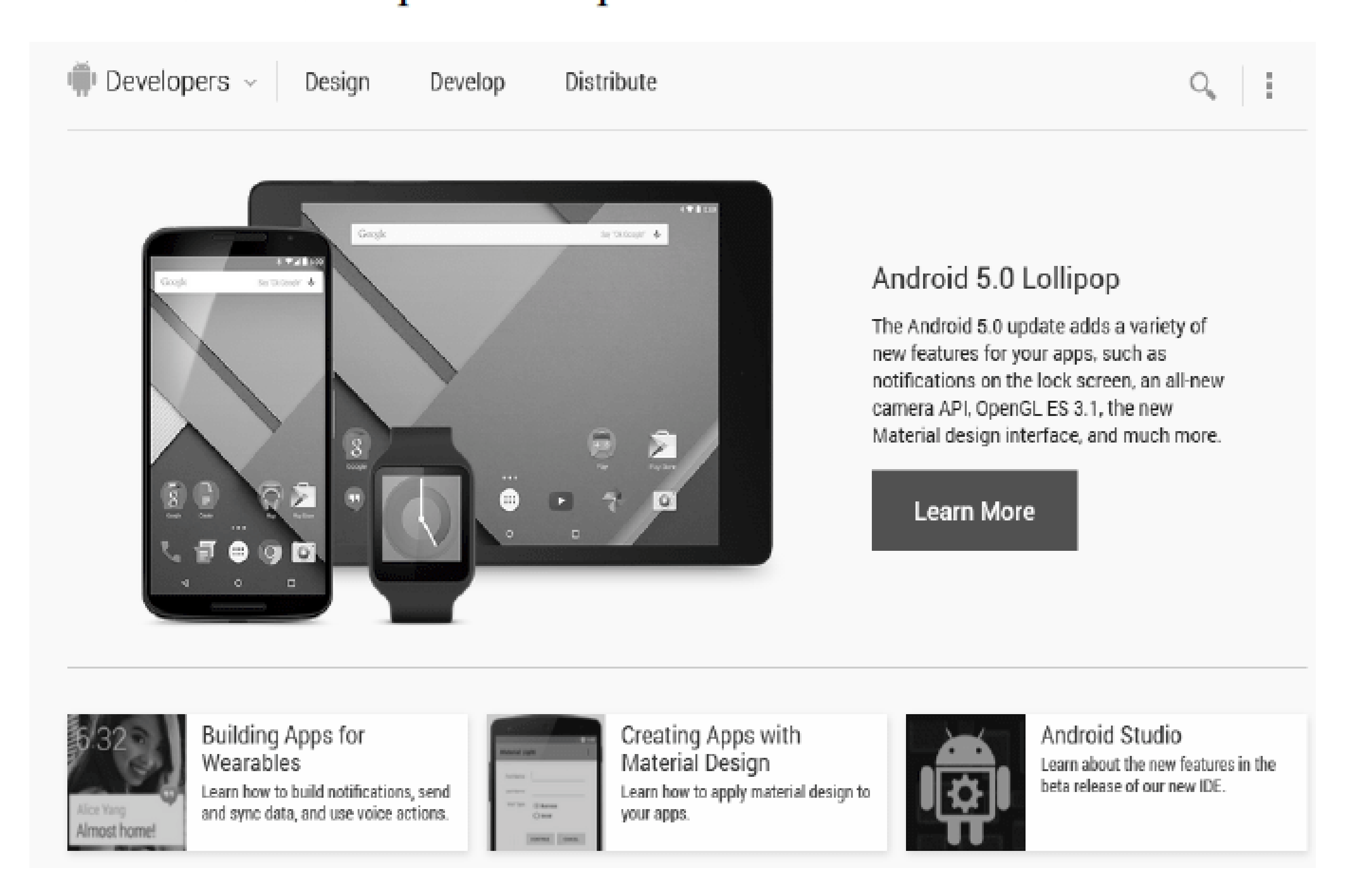


图 3-14　Android 的官方网站

（2）单击页面中部的 Get the SDK 链接，如图 3-15 所示。



图 3-15　单击 Get the SDK 链接

（3）在弹出的新页面中单击 Download the SDK 按钮，如图 3-16 所示。

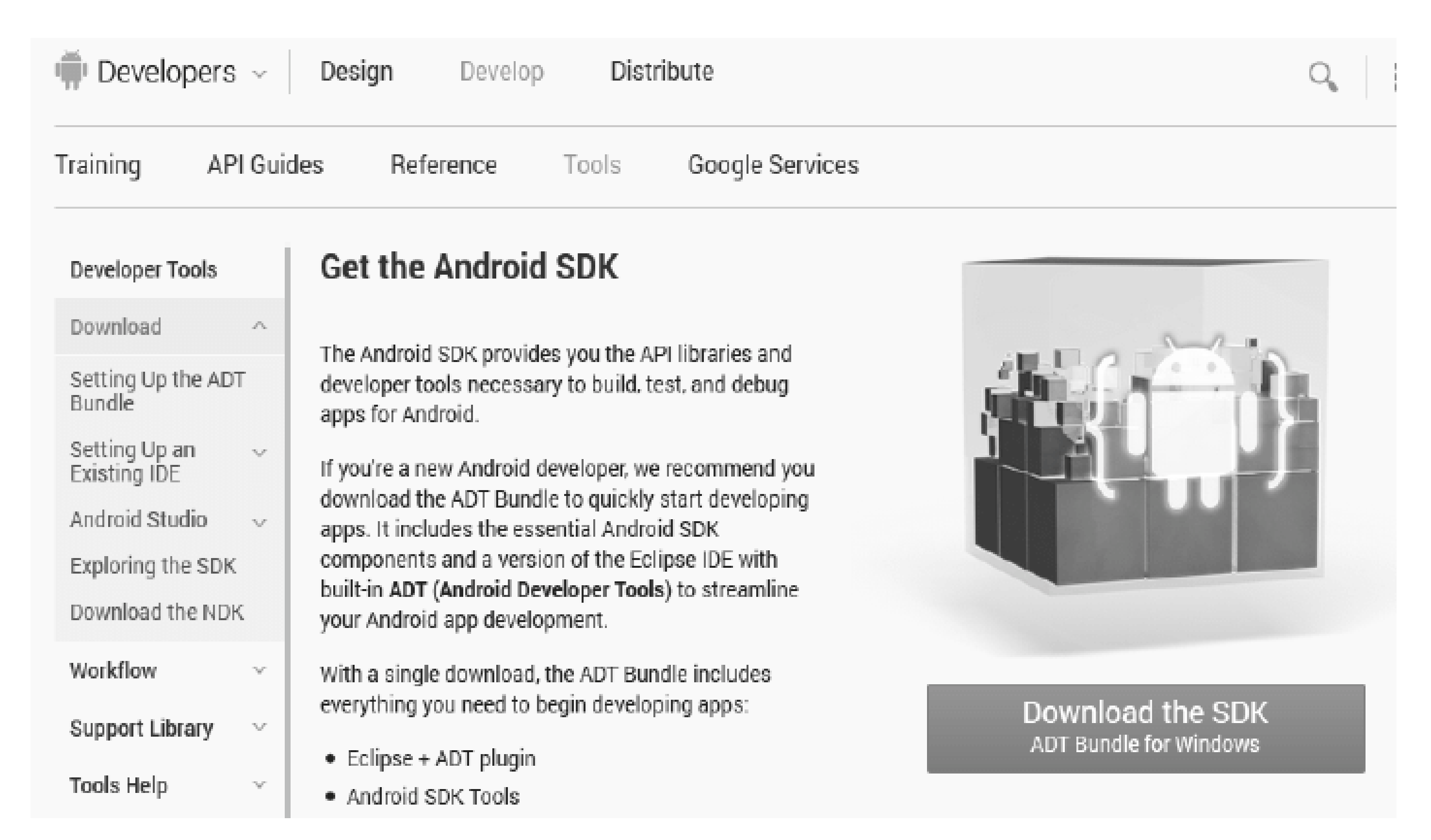
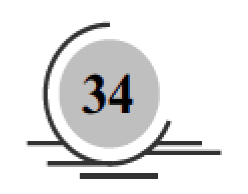


图 3-16　单击 Download the SDK 按钮



（4）在弹出的 Get the Android SDK 界面中选中 I have read and agree with the above terms and conditions 复选框，然后在下面的单选按钮中选择系统的位数。例如笔者的机器是 32 位的，所以选中 32-bit 单选按钮，如图 3-17 所示。

（5）单击图 3-17 中的 Download the SDK ADT Bundle for Windows 按钮后开始下载工作，下载的目标文件是一个压缩包，如图 3-18 所示。

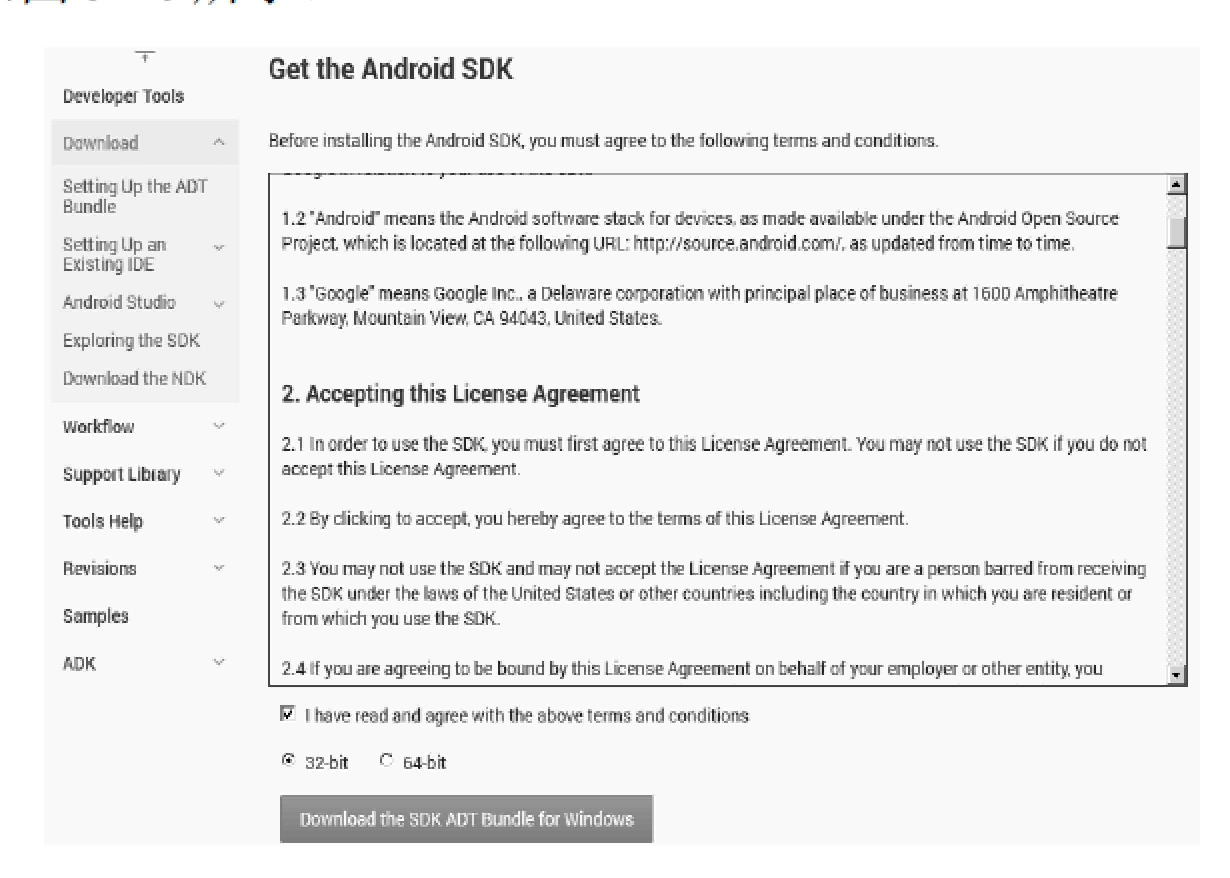




图 3-17　Get the Android SDK 界面　　　　图 3-18　开始下载目标文件压缩包

（6）将下载得到的压缩包进行解压，解压后的目录结构如图 3-19 所示。

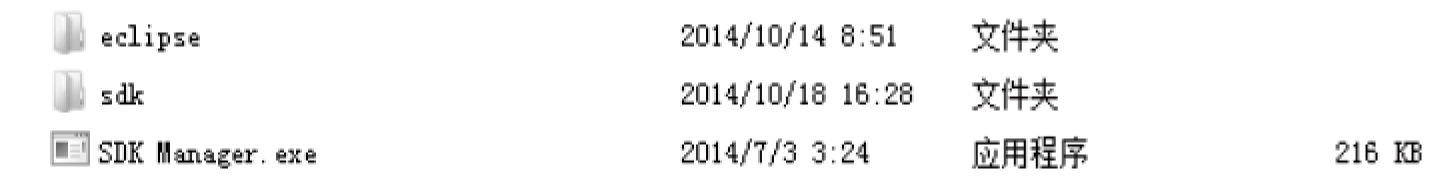


图 3-19　解压后的目录结构

由此可见，Android 官方已经将 Eclipse 和 Android SDK 实现了集成。双击 eclipse 目录中的 eclipse.exe 可以打开 Eclipse，界面效果如图 3-20 所示。

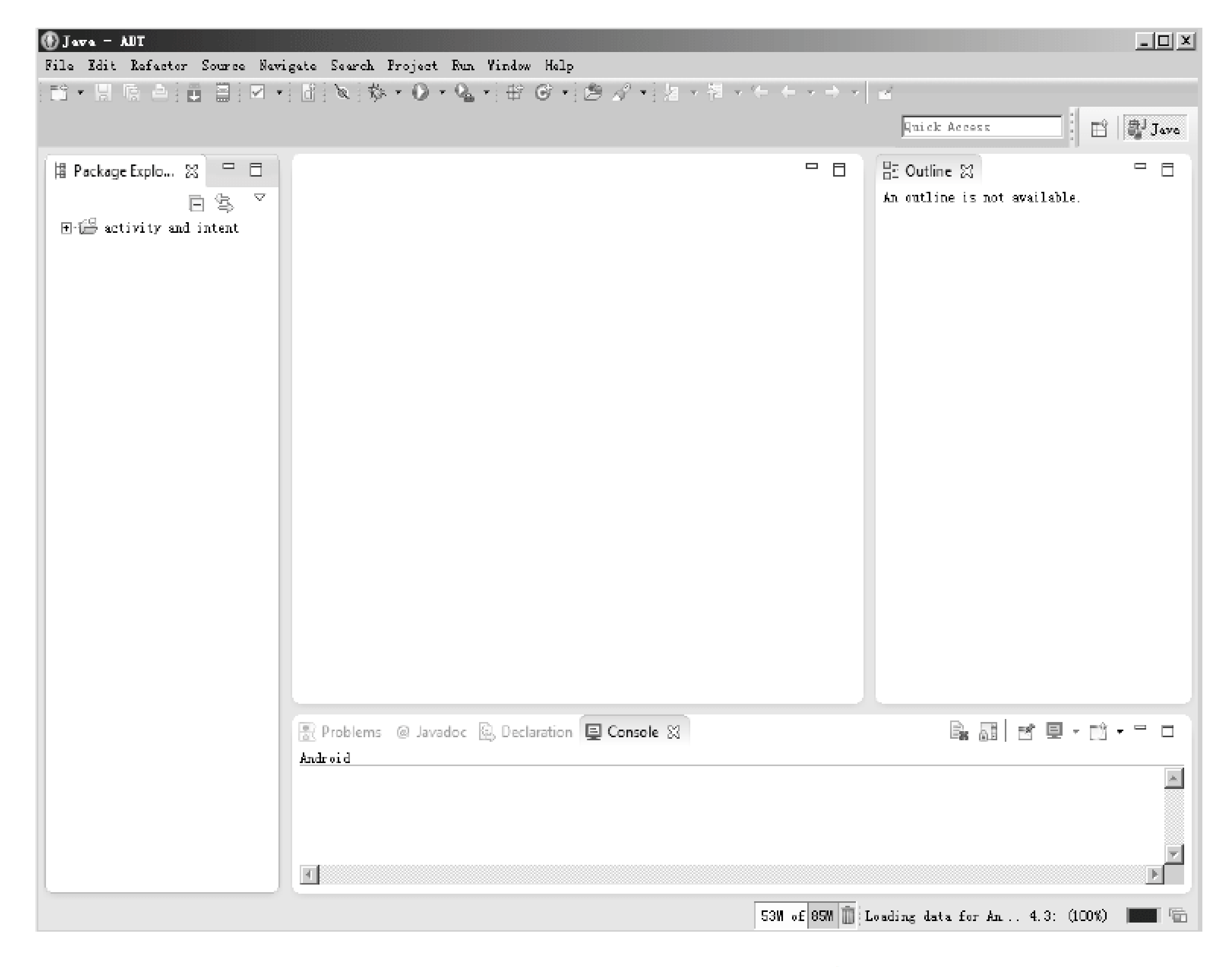


图 3-20　打开 Eclipse 后的界面效果

（7）打开 Android SDK 的方法有两种，第一种是双击下载目录中的 SDK Manager.exe 文件，第二种是在 Eclipse 工具栏中单击图标。打开后的效果如图 3-21 所示。

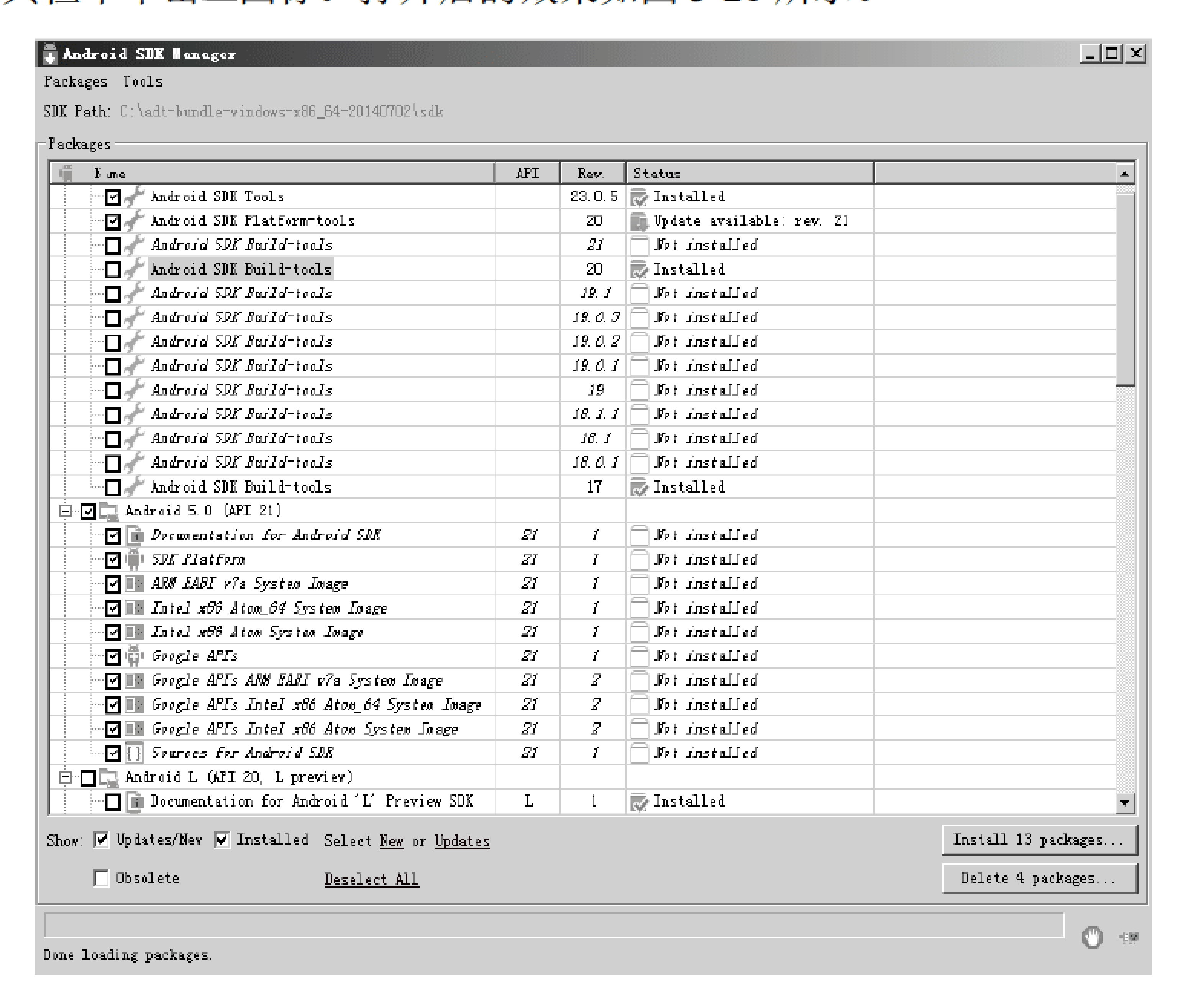


图 3-21　打开 Android SDK 后的界面效果

在打开 Android SDK 后的界面中可知，当前最新的 Android SDK 版本是 5.0。

注意：快速安装 SDK 通过 Android SDK Manager 在线安装的速度非常慢，而且有时容易挂掉。其实我们可以先从网络中寻找到 SDK 资源，用迅雷等下载工具下载后，将其放到指定目录后就可以完成安装。具体方法是先下载 android-sdk-windows（可以更新的那种），然后在 android-sdk-windows 下双击 setup.exe，在更新的过程中会发现安装 Android SDK 的速度是 1Kib/s，此时打开迅雷，分别输入下面的地址：

https://dl-ssl.google.com/android/repository/platform-tools_r05-windows.zip
https://dl-ssl.google.com/android/repository/docs-3.1_r03-linux.zip
https://dl-ssl.google.com/android/repository/android-2.2_r03-windows.zip
https://dl-ssl.google.com/android/repository/android-2.3.3_r03-linux.zip
https://dl-ssl.google.com/android/repository/android-2.1_r03-windows.zip
https://dl-ssl.google.com/android/repository/samples-2.3.3_r03-linux.zip
https://dl-ssl.google.com/android/repository/samples-2.2_r03-linux.zip
https://dl-ssl.google.com/android/repository/samples-2.1_r03-linux.zip
https://dl-ssl.google.com/android/repository/compatibility_r02.zip
https://dl-ssl.google.com/android/repository/tools_r13-windows.zip
https://dl-ssl.google.com/android/repository/google_apis-10_r02.zip
https://dl-ssl.google.com/android/repository/android-2.3.1_r03-linux.zip
https://dl-ssl.google.com/android/repository/usb_driver_r04-windows.zip
https://dl-ssl.google.com/android/repository/googleadmobadssdkandroid-4.1.0.zip
https://dl-ssl.google.com/android/repository/market_licensing-r01.zip
https://dl-ssl.google.com/android/repository/market_billing_r01.zip
https://dl-ssl.google.com/android/repository/google_apis-8_r02.zip
https://dl-ssl.google.com/android/repository/google_apis-7_r01.zip

https://dl-ssl.google.com/android/repository/google_apis-9_r02.zip
…
可以继续根据自己的开发要求选择不同版本的 API

下载完后将它们复制到 android-sdk-windows/Temp 目录下，然后再运行 setup.exe，选中需要的 API 选项，会发现马上就安装好了。记得把原始文件保留好，因为放在 temp 目录下的文件装好后立刻就没有了。

3.4　安装 ADT

知识点讲解：光盘:视频\知识点\第 3 章\安装 ADT.avi

Android 为 Eclipse 定制了一个专用插件 Android Development Tools（ADT），此插件为用户提供了一个强大的开发 Android 应用程序的综合环境。ADT 扩展了 Eclipse 的功能，可以让用户快速地建立 Android 项目，创建应用程序界面。要安装 Android Development Tools plug-in，需要首先打开 Eclipse IDE，然后进行如下操作：

（1）打开 Eclipse 后，在菜单栏中选择 Help | Install New Software 命令，如图 3-22 所示。

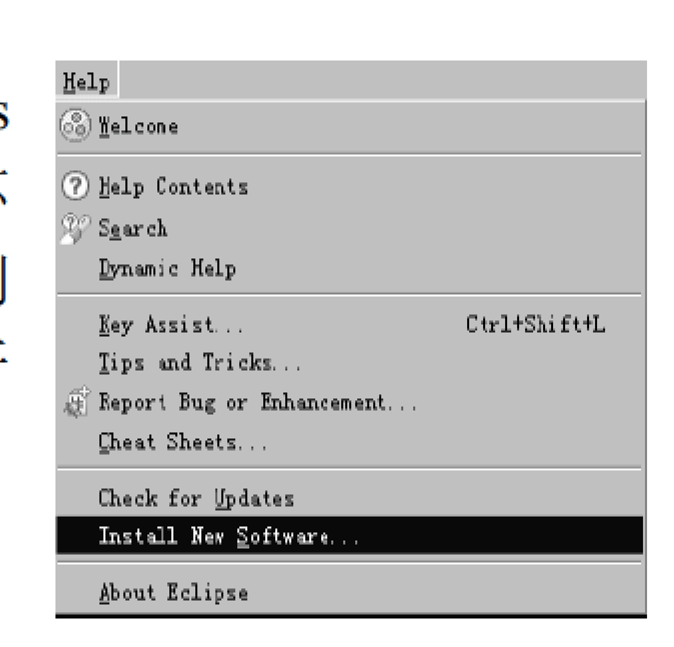


图 3-22　添加插件

（2）在弹出的对话框中单击 Add 按钮，如图 3-23 所示。

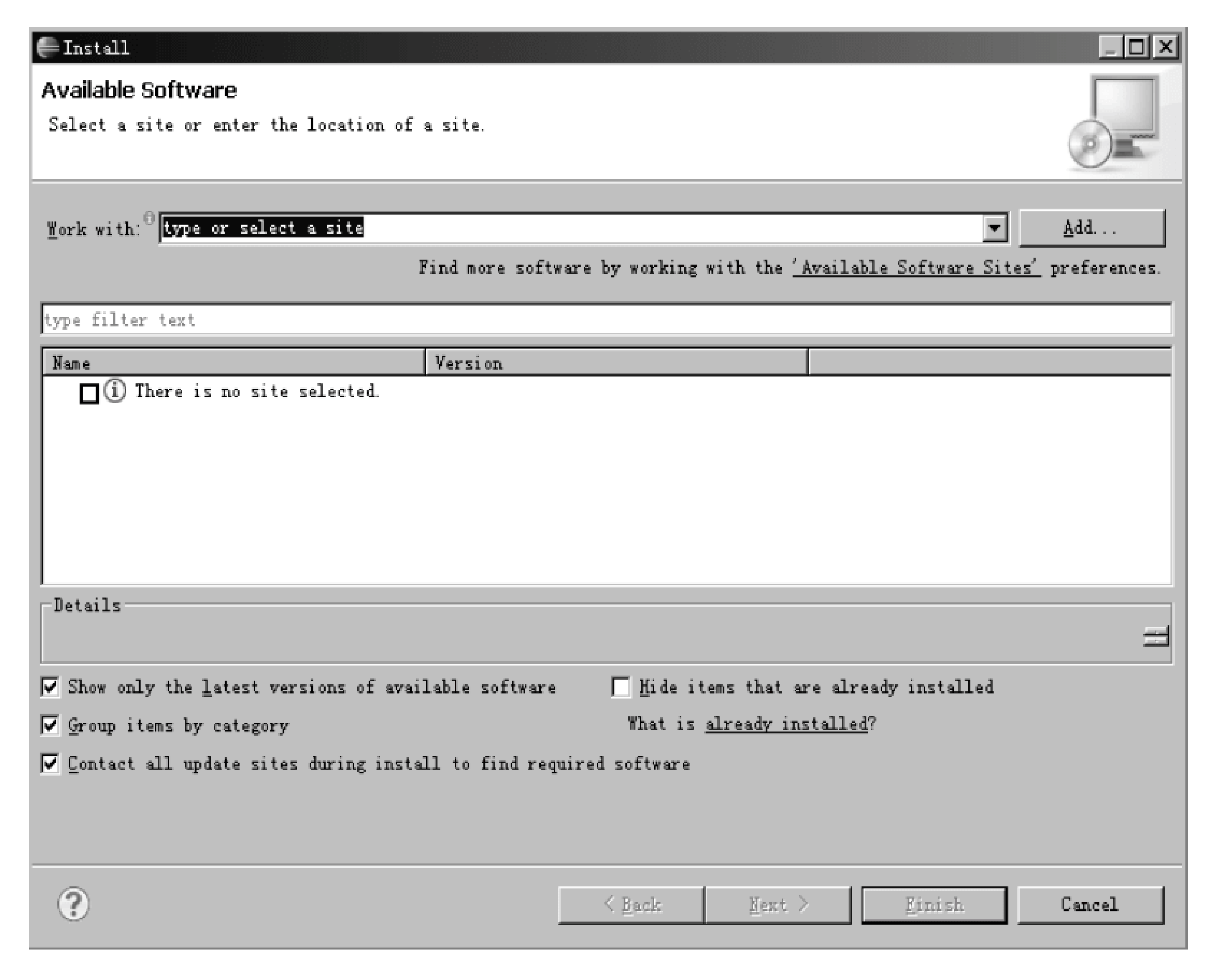


图 3-23　添加插件

（3）在弹出的 Add Site 对话框中分别输入名字和地址，名字可以自己命名，例如“123”，但是在 Location 文本框中必须输入插件的网络地址 http://dl-ssl.google.com/Android/eclipse/，如图 3-24 所示。

（4）单击 OK 按钮，此时在 Install 对话框将会显示系统中可用的插件，如图 3-25 所示。

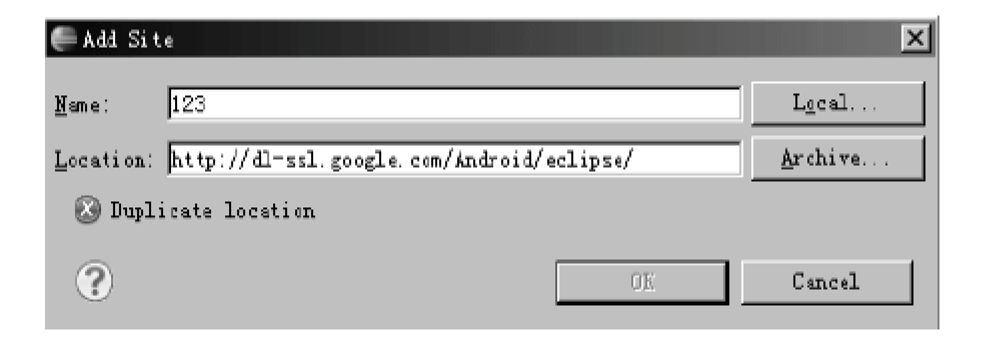


图 3-24　设置地址

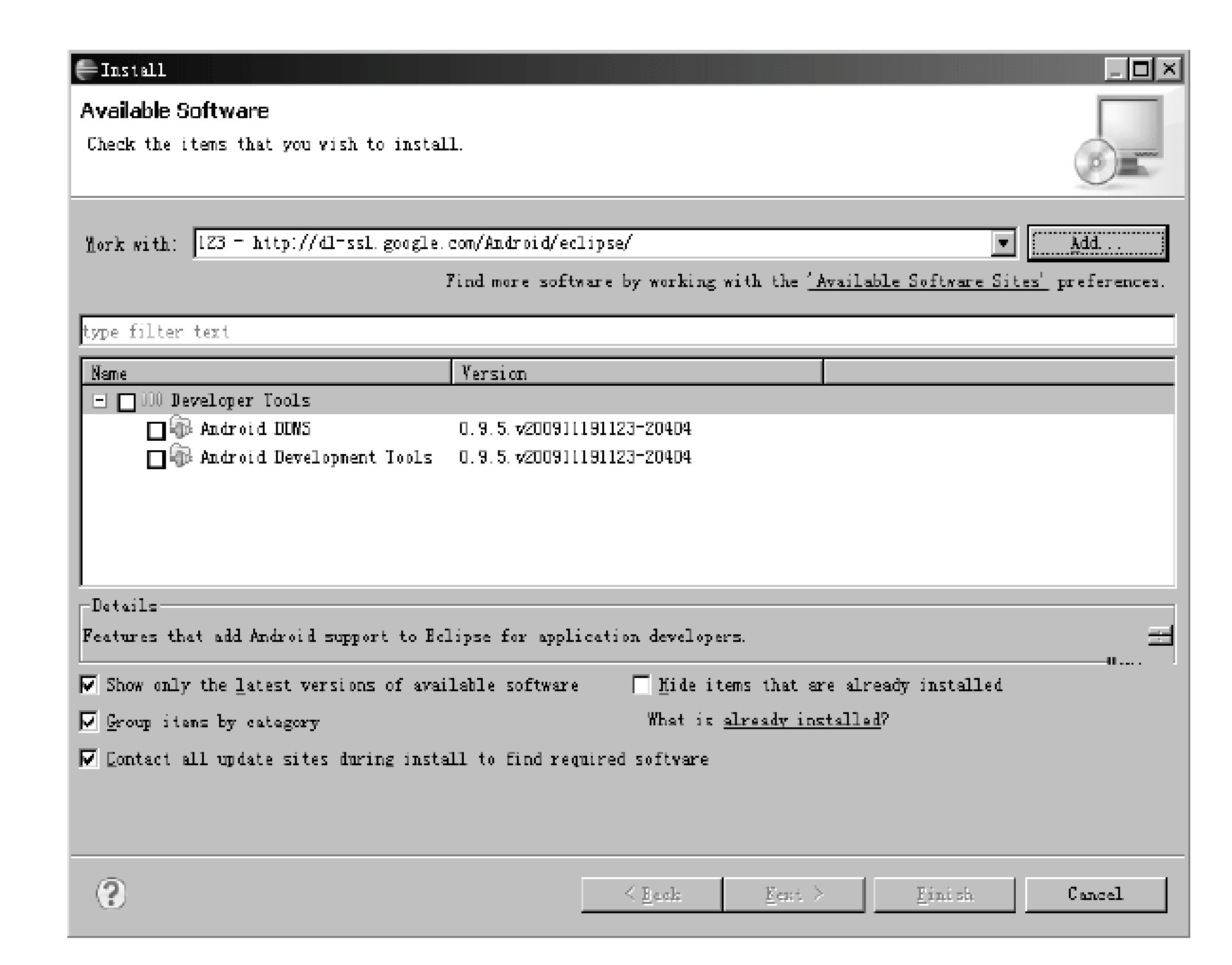


图 3-25　插件列表

（5）选中 Android DDMS 和 Android Development Tools 复选框，然后单击 Next 按钮进入安装界面，如图 3-26 所示。

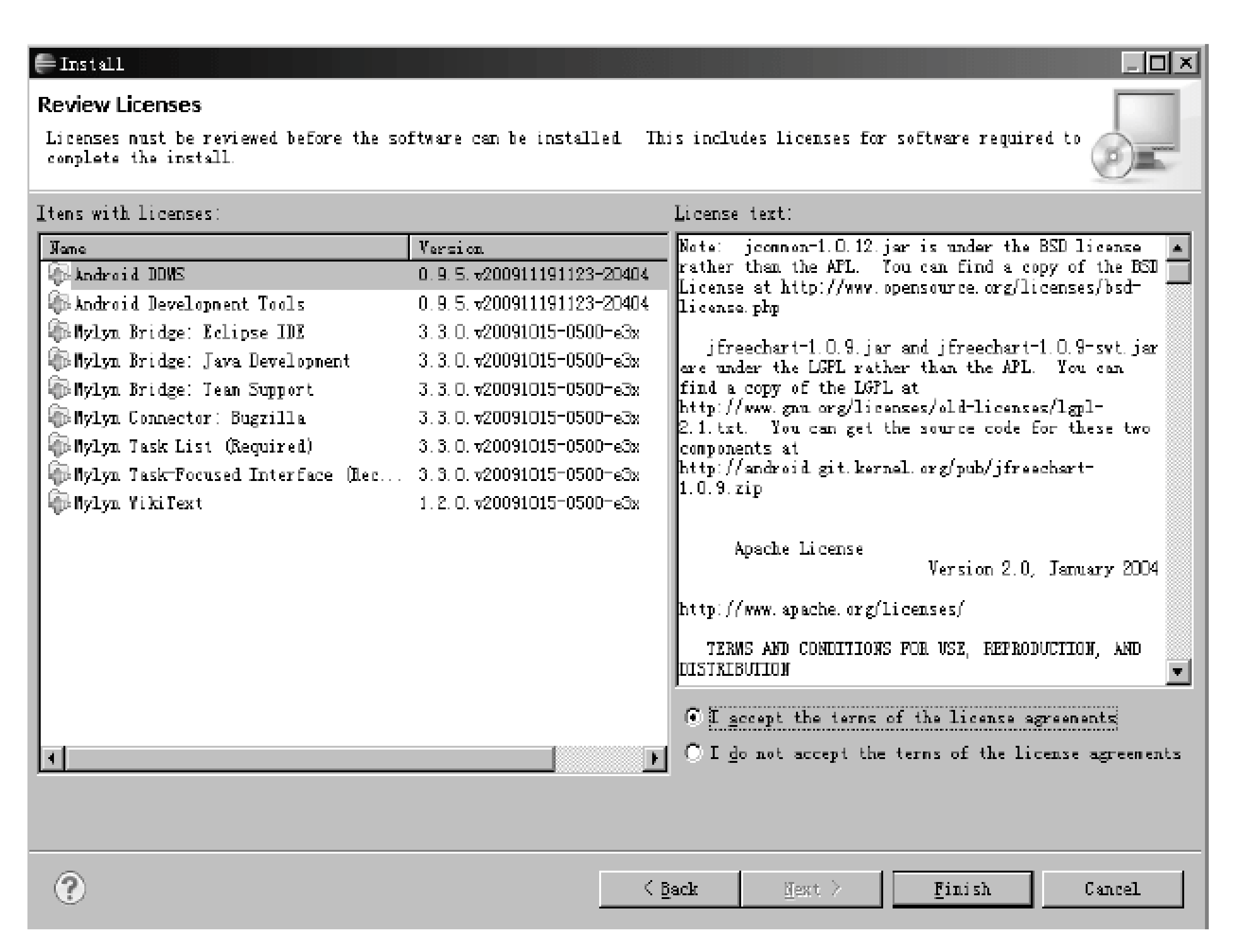


图 3-26　插件安装界面

（6）选中 I accept...单选按钮，单击 Finish 按钮，开始安装，如图 3-27 所示。

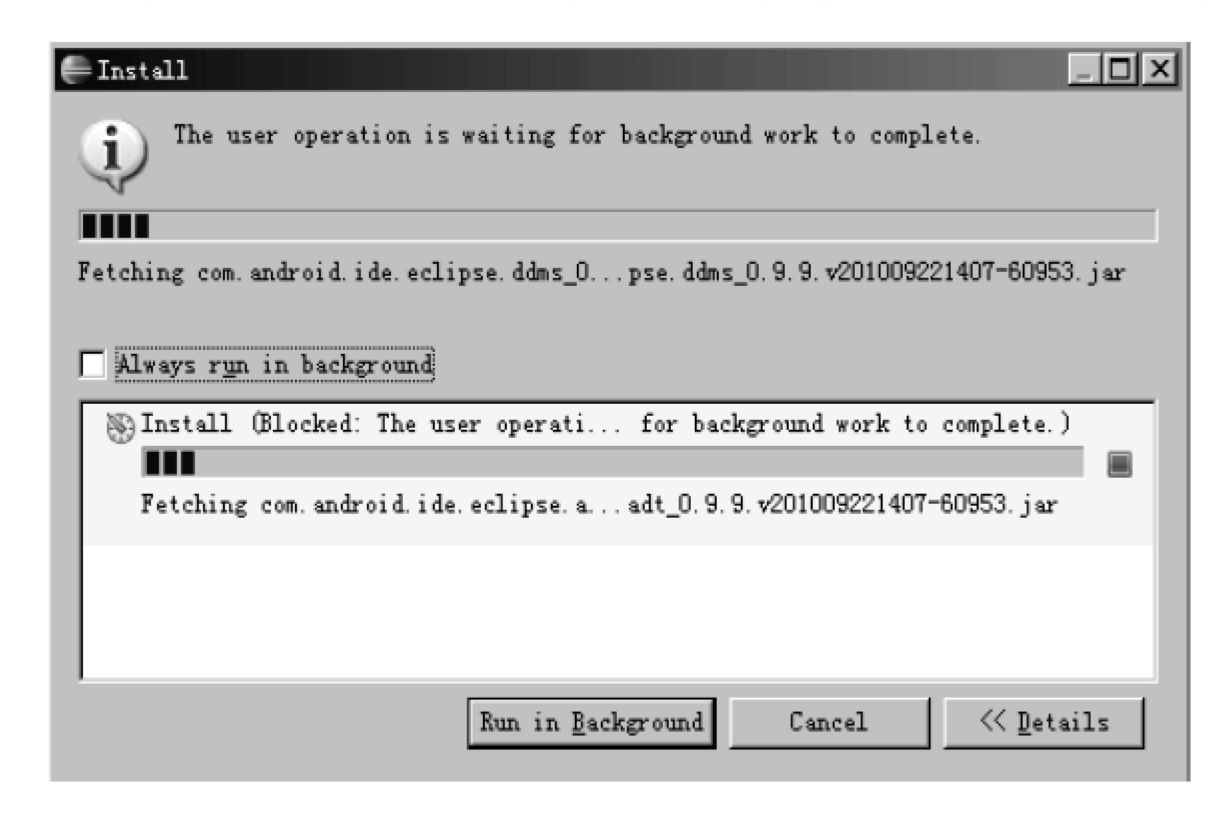
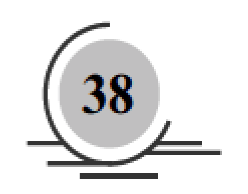


图 3-27　开始安装



注意： 在上个步骤中，可能会发生计算插件占用资源情况，过程有点慢。完成后会提示重启 Eclipse 来加载插件，等重启后就可以用了。并且不同版本的 Eclipse 安装插件的方法和步骤是不同的，但是基本上大同小异，读者可以根据操作提示自行解决。

3.5　验 证 设 置

知识点讲解：光盘:视频\知识点\第 3 章\验证设置.avi

本章前面内容已经讲解了搭建安装 Android 基本环境的知识，在完成安装之后，还需要一些具体验证和设置工作，本节将详细讲解验证和设置 Android 开发环境的基本知识。

3.5.1　设定 Android SDK Home

当完成上述插件装备工作后，此时还不能使用 Eclipse 创建 Android 项目，我们还需要在 Eclipse 中设置 Android SDK 的主目录。

（1）打开 Eclipse，在菜单中依次选择 Windows | Preferences 命令，如图 3-28 所示。

（2）在弹出的界面左侧可以看到 Android 项，选中 Android 项后，在右侧设定 Android SDK 所在目录为 SDK Location，单击 OK 按钮完成设置，如图 3-29 所示。

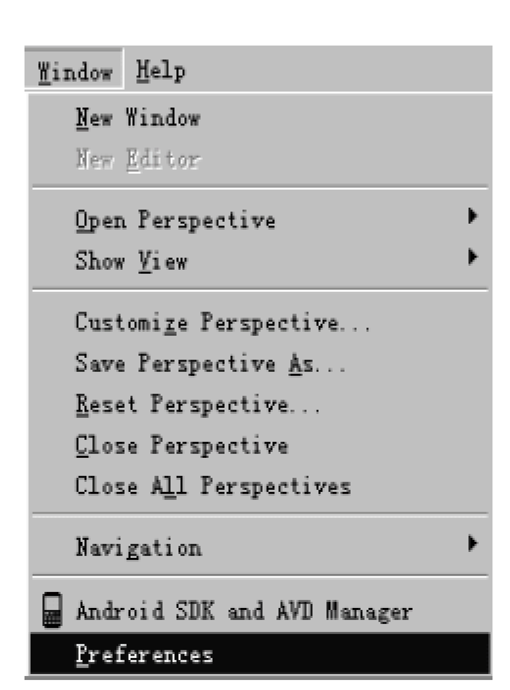


图 3-28　选择 Preferences 命令

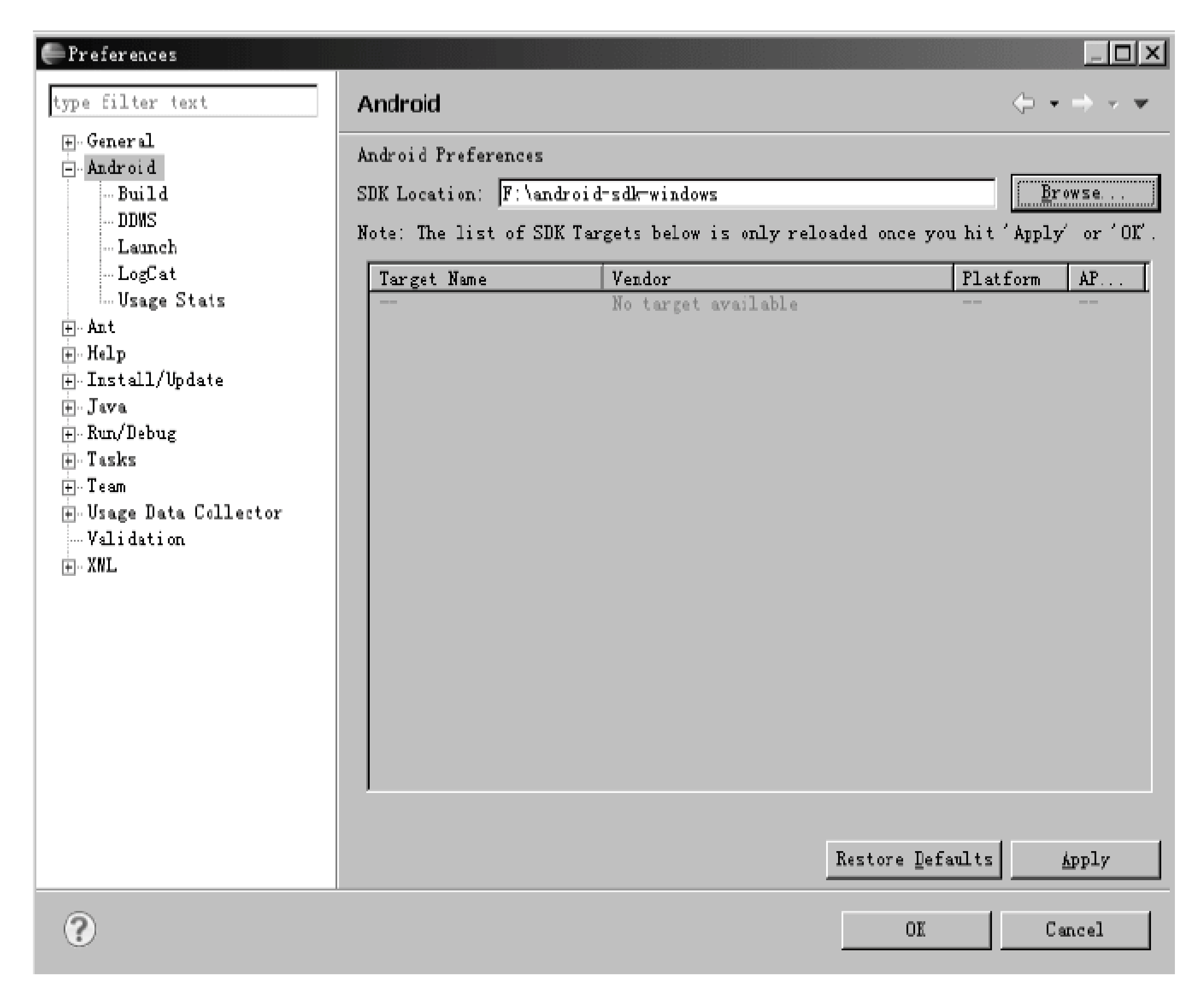


图 3-29　设定 Android SDK 所在目录

3.5.2　验证开发环境

经过前面步骤的讲解，一个基本的 Android 开发环境算是搭建完成了。都说实践是检验真理的唯一标准，下面通过新建一个项目来验证当前的环境是否可以正常工作。

（1）打开 Eclipse，在菜单中依次选择 File | New | Project 命令，在弹出的对话框中可以看到 Android 类型的选项，如图 3-30 所示。

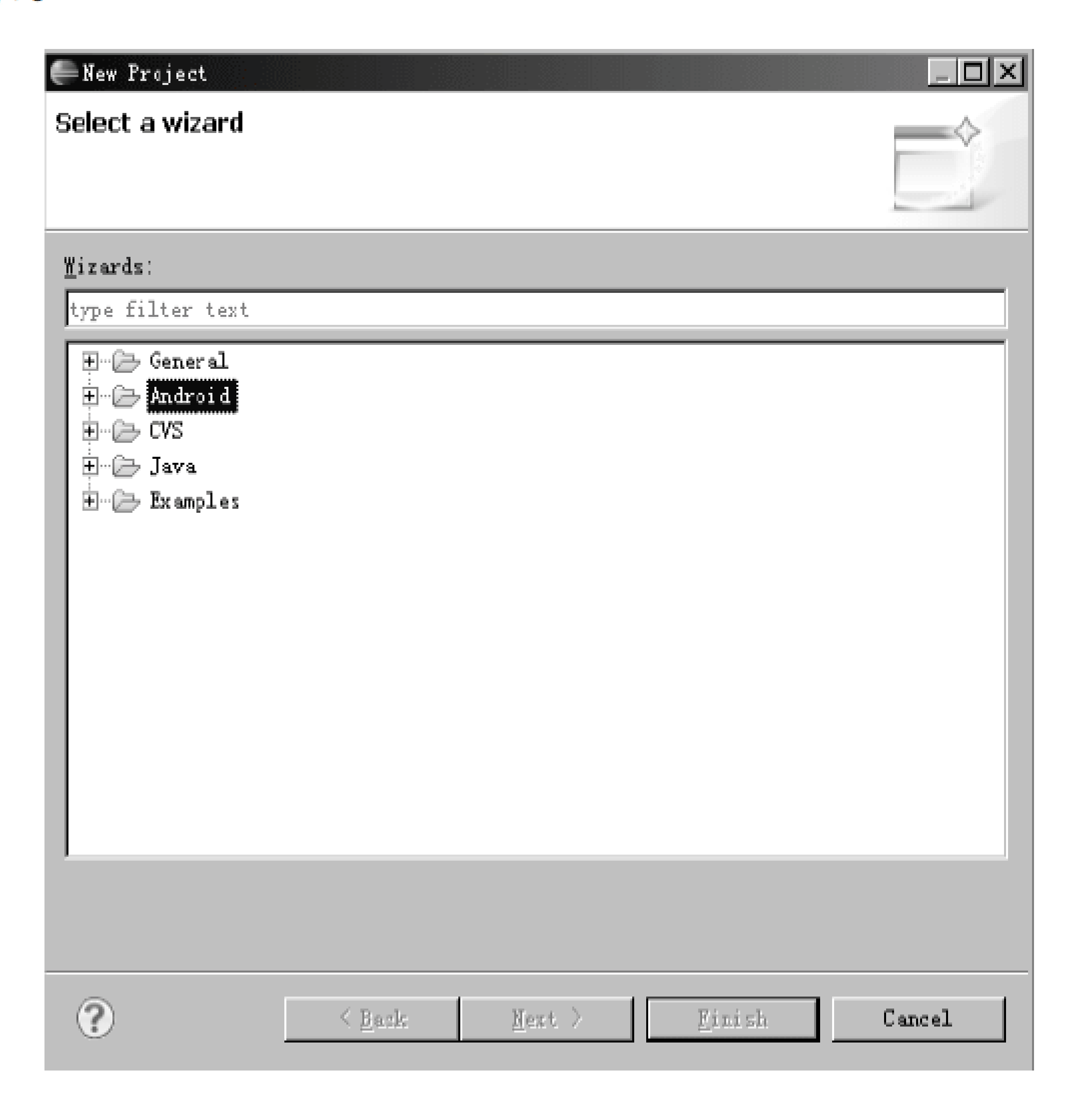


图 3-30　新建项目

（2）在图 3-30 中选择 Android 选项，单击 Next 按钮后打开 New Android Application 对话框，在对应的文本框中输入必要的信息，如图 3-31 所示。

（3）单击 Finish 按钮后 Eclipse 会自动完成项目的创建工作，最后会看到如图 3-32 所示的项目结构。

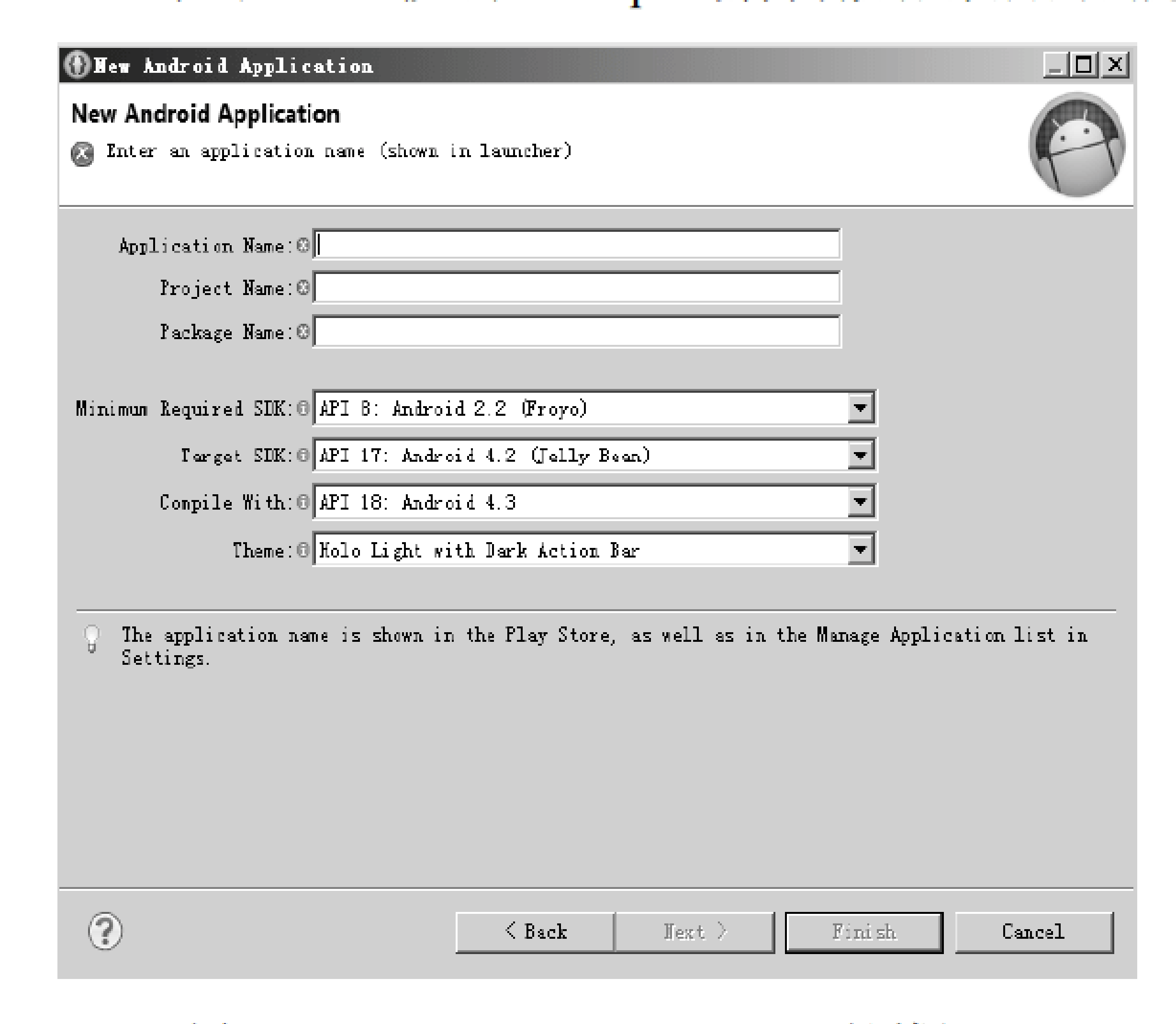


图 3-31　New Android Application 对话框

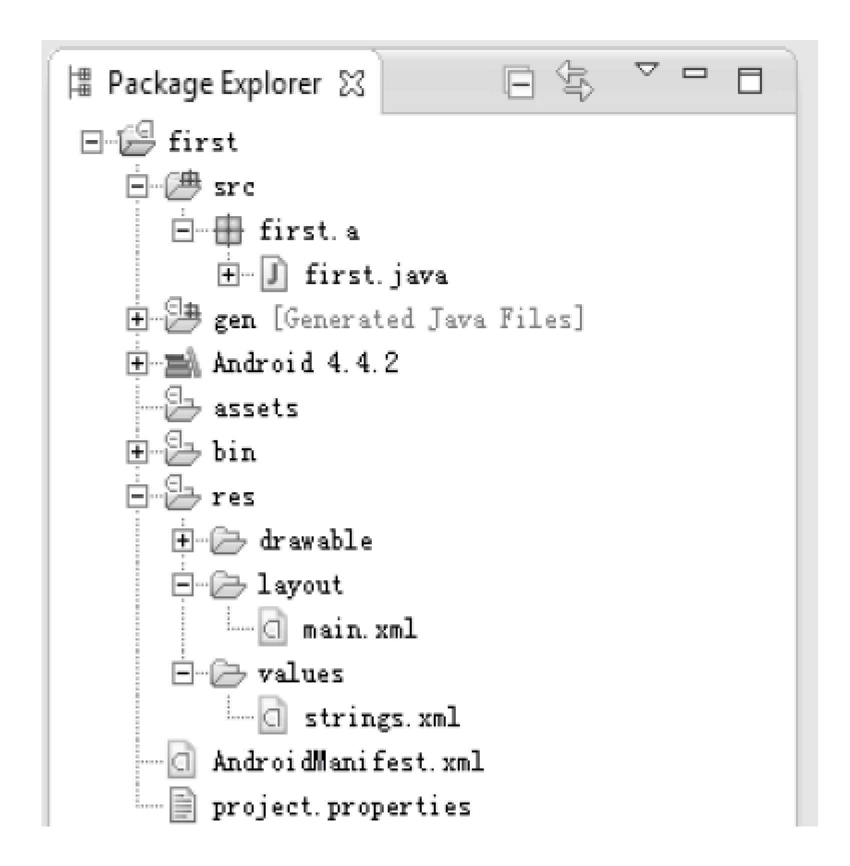
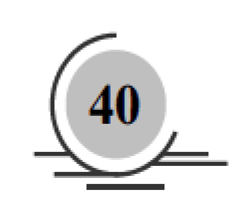


图 3-32　项目结构

3.6　Android 模拟器详解

知识点讲解：光盘:视频\知识点\第 3 章\Android 模拟器详解.avi

我们都知道程序开发需要调试，只有经过调试之后才能知道我们的程序是否正确运行。作为一款手机系统，我们怎么样才能在电脑平台上调试 Android 程序呢？不用担心，谷歌为我们提供了模拟器



来解决我们担心的问题。所谓模拟器，就是指在电脑上模拟安卓系统，可以用这个模拟器来调试并运行开发的 Android 程序。开发人员不需要一个真实的 Android 手机，只通过电脑即可模拟运行一个手机，即可开发出应用在手机上面的程序。

3.6.1　创建 Android 虚拟设备（AVD）

AVD 全称为 Android 虚拟设备（Android Virtual Device），每个 AVD 模拟了一套虚拟设备来运行 Android 平台，这个平台至少要有自己的内核、系统图像和数据分区，还可以有自己的 SD 卡和用户数据以及外观显示等。

Android 模拟器不能完全替代真机，具体来说有如下差异。

- 模拟器不支持呼叫和接听实际来电，但可以通过控制台模拟电话呼叫（呼入和呼出）。
- 模拟器不支持USB连接。
- 模拟器不支持相机/视频捕捉。
- 模拟器不支持音频输入（捕捉），但支持输出（重放）。
- 模拟器不支持扩展耳机。
- 模拟器不能确定连接状态。
- 模拟器不能确定电池电量水平和交流充电状态。
- 模拟器不能确定SD卡的插入/弹出。
- 模拟器不支持蓝牙。

创建 AVD 的基本步骤如下。

（1）单击 Eclipse 菜单中的图标，如图 3-33 所示。

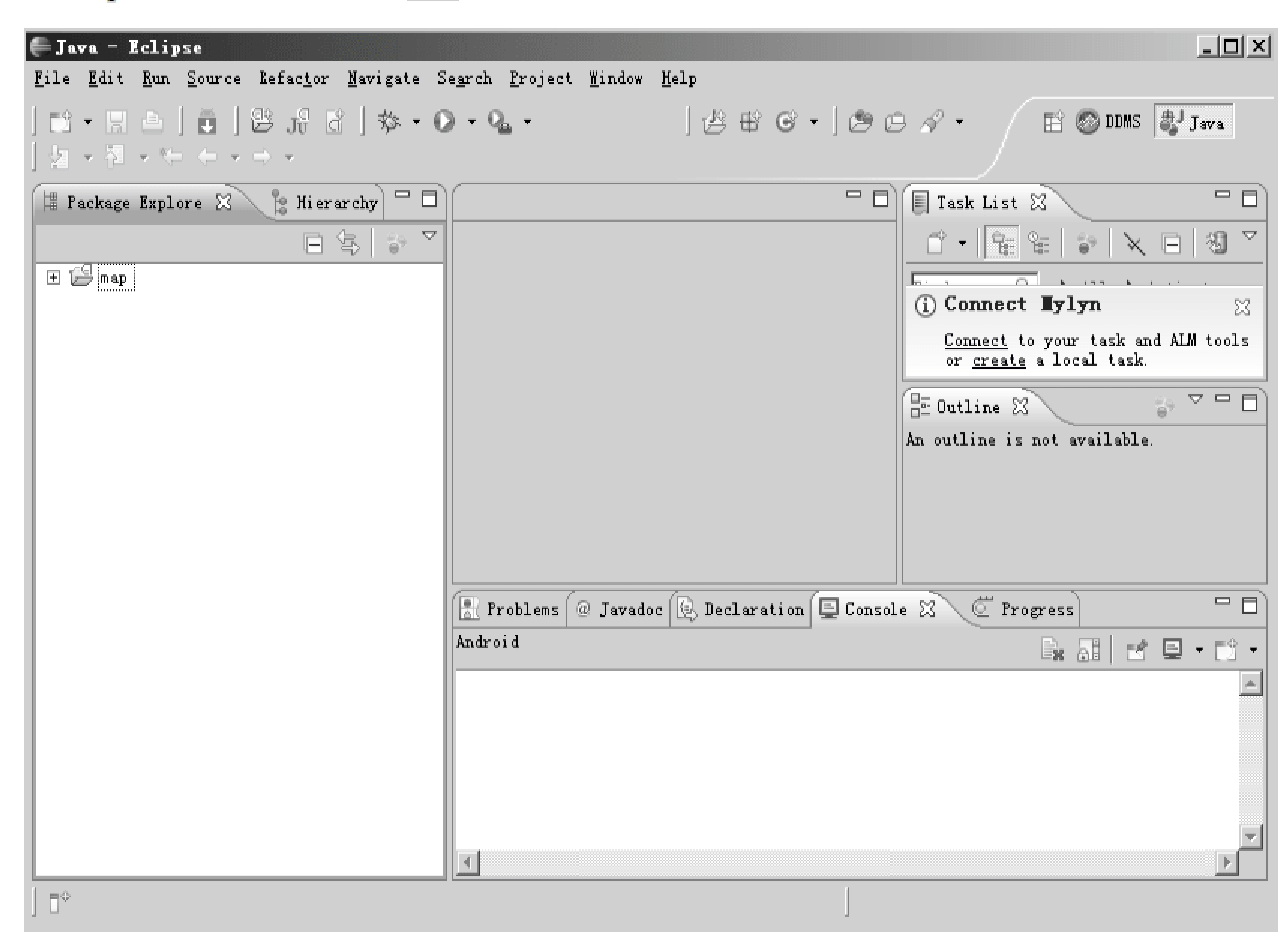


图 3-33　Eclipse 窗口

（2）在弹出的 Android Virtual Device（AVD）Manager 对话框中选择 Android Virtual Devices 选项卡，如图 3-34 所示。

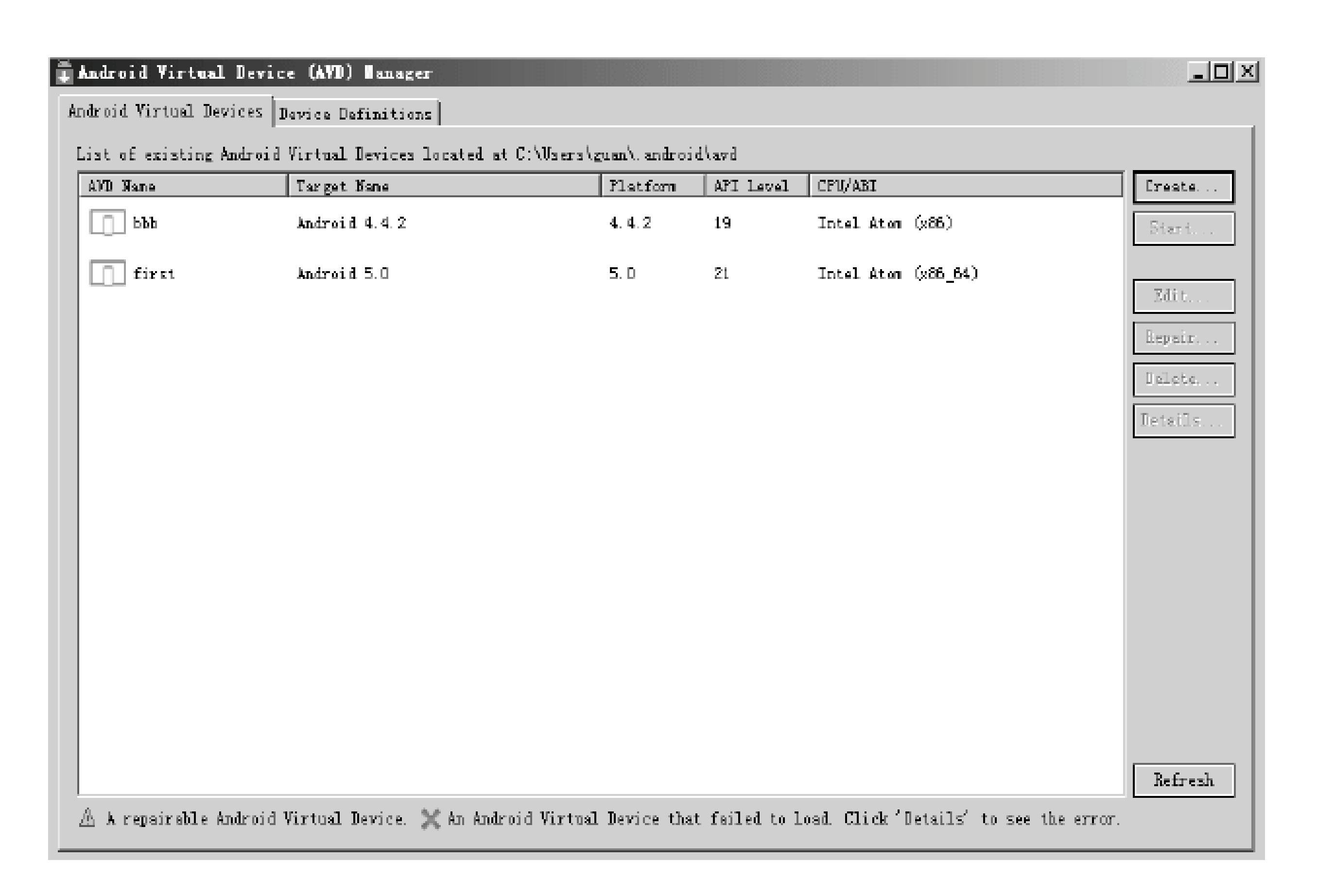


图 3-34　Android Virtual Device（AVD）Manager 对话框

在其中的列表框中列出了当前已经安装的 AVD 版本，我们可以通过右侧的按钮来创建、删除或修改 AVD。主要按钮的具体说明如下。

- Create...：创建一个新的AVD，单击此按钮在弹出的界面中可以创建一个新AVD，如图3-35所示。
- Edit...：修改已经存在的AVD。
- Delete...：删除已经存在的AVD。
- Start...：启动一个AVD模拟器。

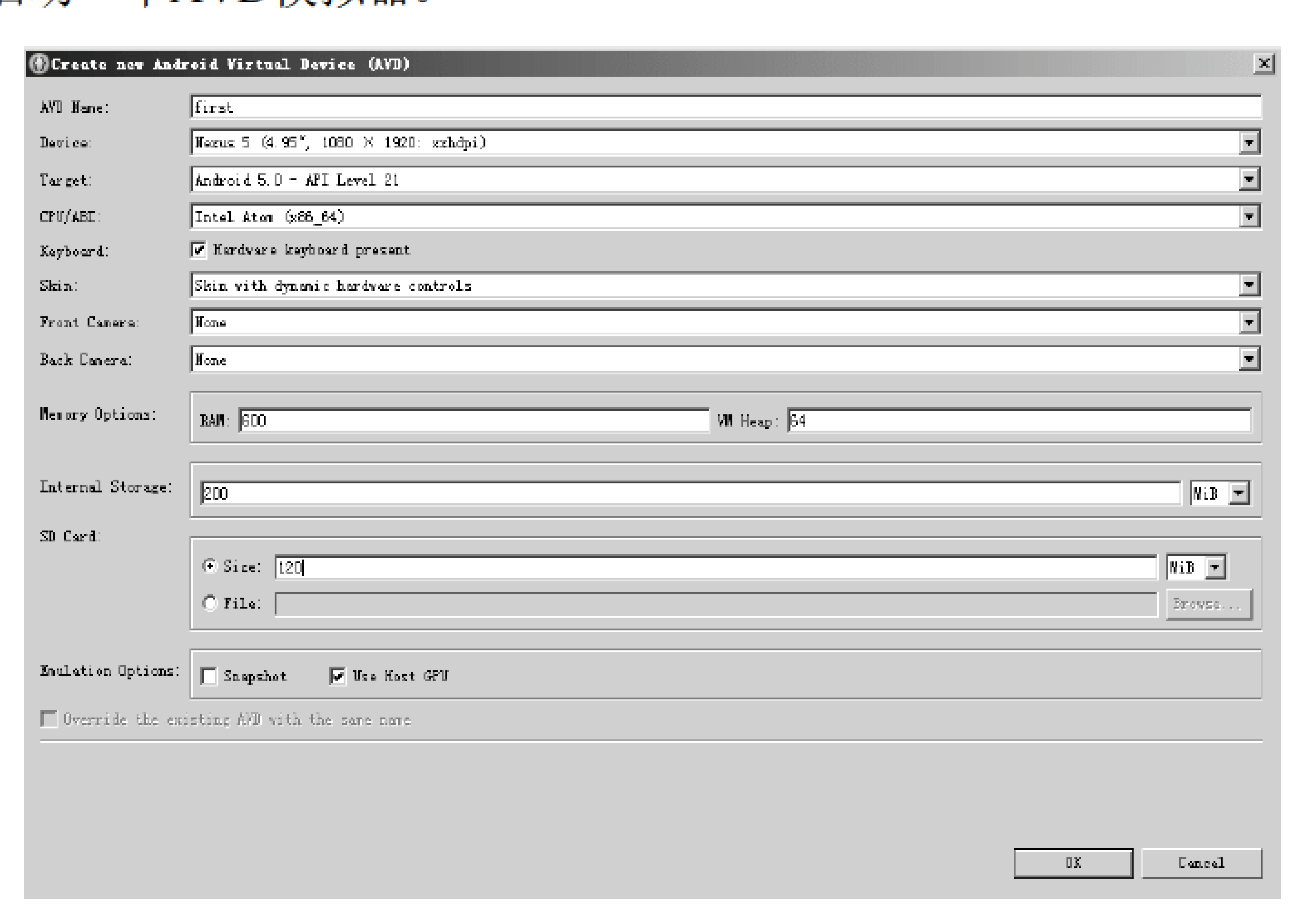


图 3-35　新建 AVD 界面

注意：我们可以在 CMD 中创建或删除 AVD，例如可以按照如下 CMD 命令创建一个 AVD。

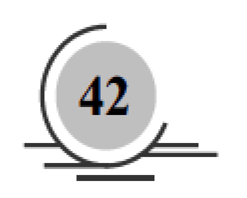
```
android create avd --name <your_avd_name> --target <targetID>
```

其中 your_avd_name 是需要创建的 AVD 的名字，CMD 窗口如图 3-36 所示。

```
C:\WINDOWS\system32\cmd.exe - android create avd --name mm --target 2
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>android create avd --name mm --target 2
Android 1.5 is a basic Android platform.
Do you wish to create a custom hardware profile [no]
```

图 3-36　CMD 窗口



3.6.2　启动 AVD 模拟器

对于 Android 程序的开发者来说，模拟器的推出给开发者在开发和测试上带来了很大的便利。无论在 Windows 下还是 Linux 下，Android 模拟器都可以顺利运行，并且官方提供了 Eclipse 插件，可以将模拟器集成到 Eclipse 的 IDE 环境。Android SDK 中包含的模拟器的功能非常齐全，电话本、通话等功能都可正常使用（当然你没办法真的从这里打电话），甚至其内置的浏览器和 Maps 都可以联网。用户可以使用键盘输入，鼠标单击模拟器按键输入，甚至还可以使用鼠标单击、拖动屏幕进行操纵。模拟器在电脑上模拟运行的效果如图 3-37 所示。



图 3-37　模拟器

在调试的时候我们需要启动 AVD 模拟器，启动 AVD 模拟器的基本流程如下。

（1）选择图 3-34 列表中名为 first 的 AVD，单击 Start... 按钮后弹出 Launch Options 对话框，如图 3-38 所示。

（2）单击 Launch 按钮后将会运行名为 first 的模拟器，运行界面效果如图 3-39 所示。

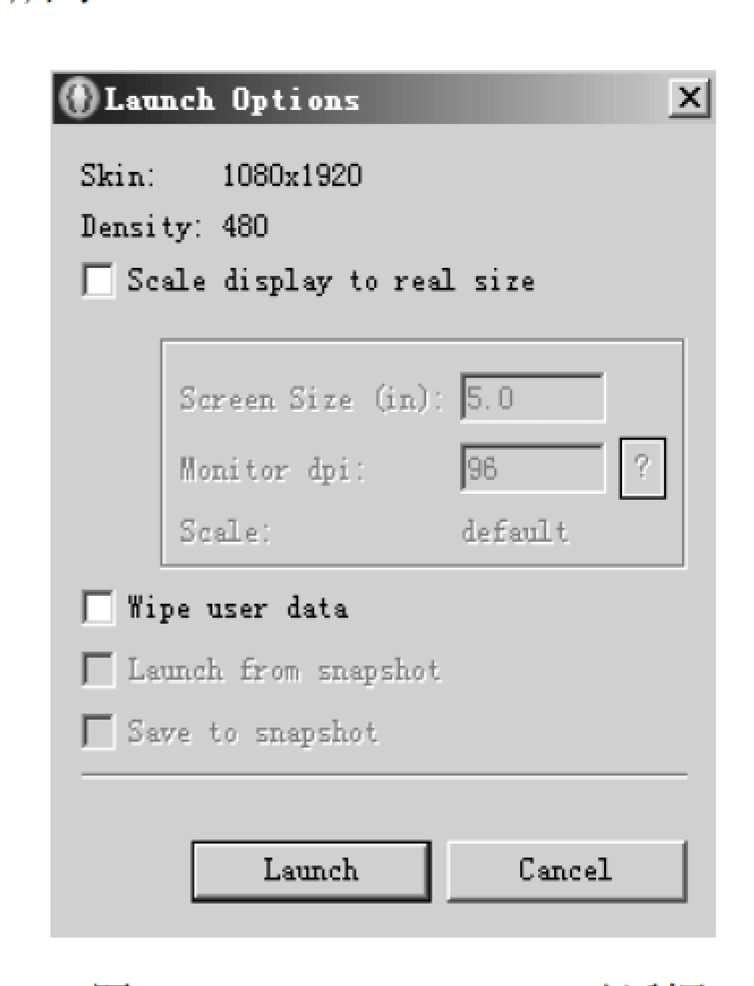


图 3-38　Launch Options 对话框

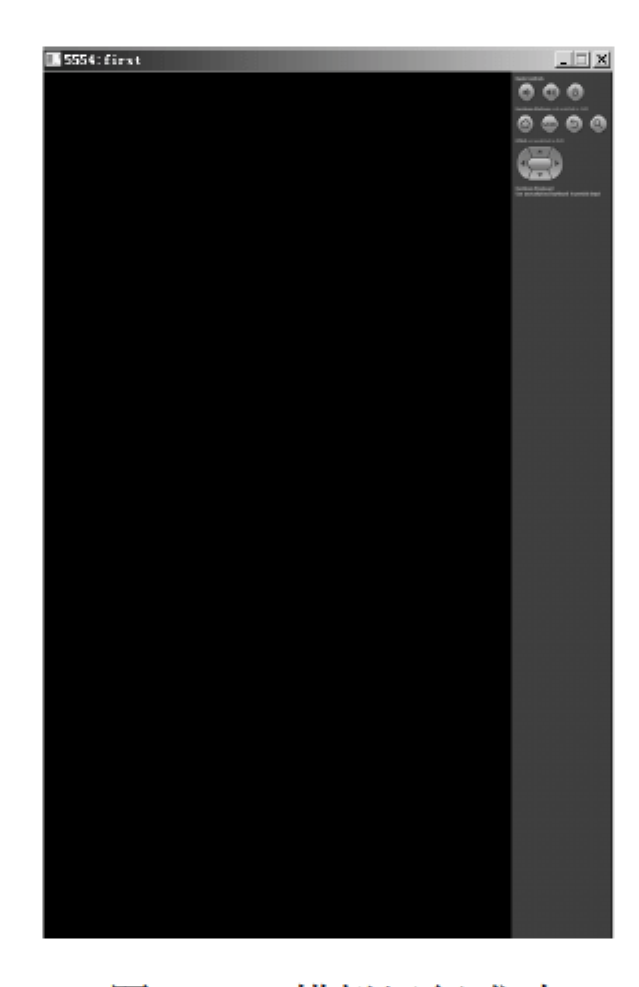


图 3-39　模拟运行成功

第 4 章　Android 核心框架详解

学习编程不能打无把握之仗，学习 Android 外设项目开发也是如此。要想真正精通开发 Android 外设项目程序技术的精髓，不但需要学习底层和 Android 框架方面的知识，而且还需要掌握 Android 顶层应用程序开发的基本知识。本章将详细讲解 Android 系统的体系结构，了解外设项目开发所必须具备的基本技术，为读者学习后面的知识打下基础。

4.1　Android 系统架构介绍

知识点讲解：光盘:视频\知识点\第 4 章\Android 系统架构介绍.avi

Android 是一个移动设备的开发平台，其软件层次结构包括操作系统（OS）、中间件（MiddleWare）和应用程序（Application）。整个 Android 系统的层次结构分为 4 层，自下而上分别如下。

- 操作系统层（OS）。
- 各种库（Libraries）和Android 运行环境（RunTime）。
- 应用程序框架（Application Framework）。
- 应用程序（Application）。

上述各个层的具体结构如图 4-1 所示。

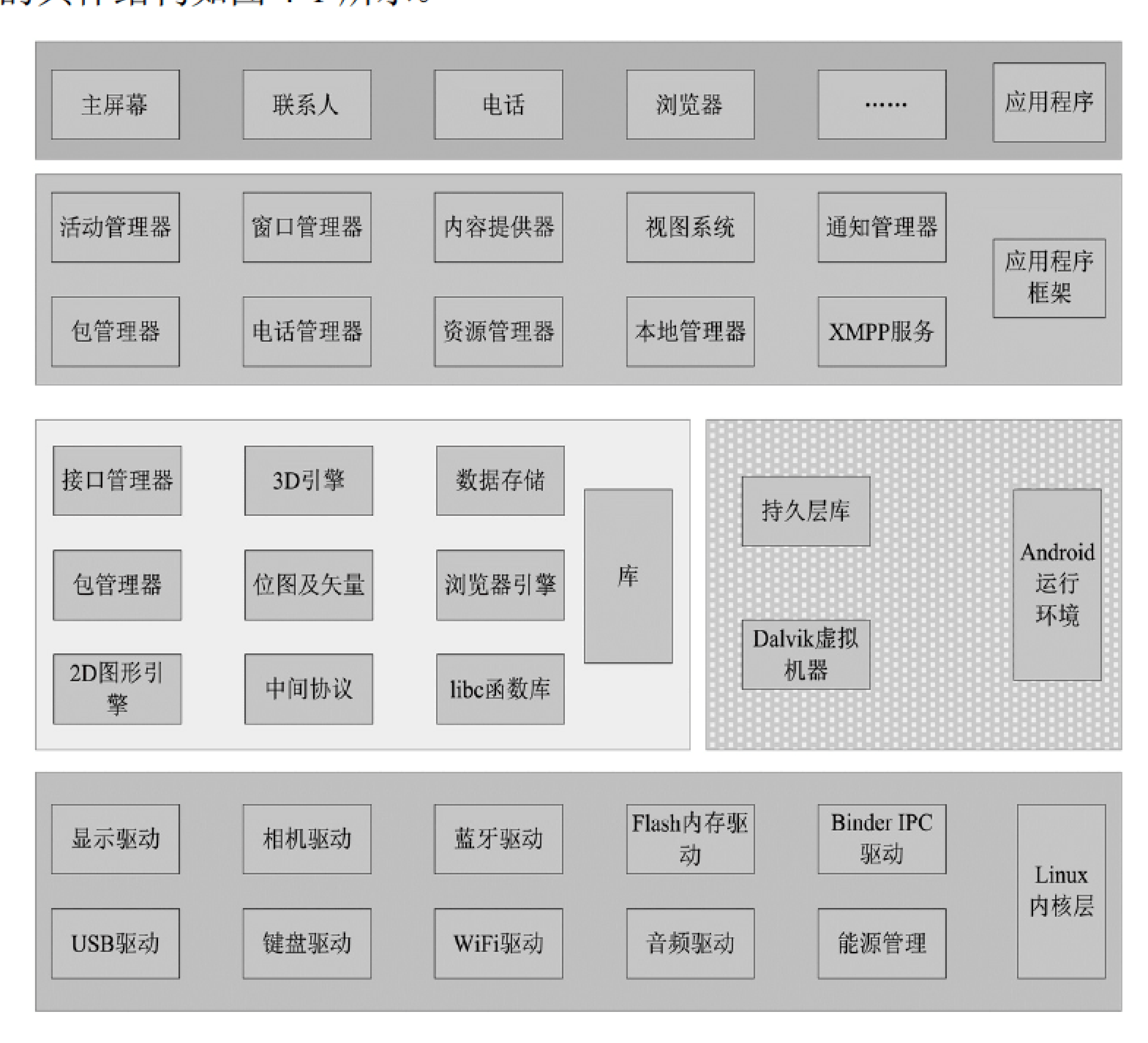


图 4-1　Android 操作系统的组件结构图

为了更加深入理解 Android 系统的技术精髓，初学者们很有必要了解 Android 系统的整体架构，了解它的具体组成。只有这样才能知道 Android 究竟能干什么，我们所要学的是什么。本节简要讲解 Android 系统架构的基本知识。

4.1.1　底层操作系统层（OS）

因为 Android 基于 Linux 内核，所以 Android 使用 Linux 作为操作系统。Linux 是一种标准的技术，Linux 也是一个开放的操作系统。Android 对操作系统的使用包括核心和驱动程序两部分，Android 的 Linux 核心为标准的 Linux 内核，Android 更多的是需要一些与移动设备相关的驱动程序。主要的驱动如下。

- 显示驱动（Display Driver）：常用基于Linux的帧缓冲（Frame Buffer）驱动。
- Flash内存驱动（Flash Memory Driver）：是基于MTD的Flash驱动程序。
- 照相机驱动（Camera Driver）：常用基于Linux的v4l（Video for）驱动。
- 音频驱动（Audio Driver）：常用基于ALSA（Advanced Linux Sound Architecture，高级Linux声音体系）驱动。
- WiFi驱动（Camera Driver）：基于IEEE 802.11标准的驱动程序。
- 键盘驱动（KeyBoard Driver）：作为输入设备的键盘驱动。
- 蓝牙驱动（Bluetooth Driver）：基于IEEE 802.15.1标准的无线传输技术。
- Binder IPC驱动：Android一个特殊的驱动程序，具有单独的设备节点，提供进程间通信的功能。
- Power Management（能源管理）：管理电池电量等信息。

4.1.2　各种库（Libraries）和 Android 运行环境（RunTime）

Android 的本层次分成两个部分，一个是各种库，另一个是 Android 运行环境。本层次对应一般嵌入式系统，相当于中间件层次。本层的内容大多是使用 C 和 C++实现的。其中包含的各种库如下。

- C库：C语言的标准库，也是系统中一个最为底层的库，C库是通过Linux的系统调用来实现。
- 多媒体框架（MediaFrameword）：这部分内容是Android多媒体的核心部分，基于PacketVideo（即PV）的OpenCORE，从功能上本库一共分为两大部分，一个部分是音频、视频的回放（PlayBack），另一部分则是音视频的记录（Recorder）。
- SGL：2D图像引擎。
- SSL：即Secure Socket Layer位于TCP/IP协议与各种应用层协议之间，为数据通信提供安全支持。
- OpenGL ES 1.0：提供对3D的支持。
- 界面管理工具（Surface Management）：提供了对管理显示子系统等功能。
- SQLite：一个通用的嵌入式数据库。
- WebKit：网络浏览器的核心。
- FreeType：位图和矢量字体的功能。

Android 的各种库一般是以系统中间件的形式提供的，它们均有的一个显著特点就是与移动设备的平台应用密切相关。

Android 运行环境主要是指的虚拟机技术——Dalvik。Dalvik 虚拟机和一般 Java 虚拟机（Java VM）不同，它执行的不是 Java 标准的字节码（Bytecode），而是 Dalvik 可执行格式（.dex）中执行文件。在

执行的过程中，每一个应用程序即一个进程（Linux 的一个 Process）。二者最大的区别在于 Java VM 是基于栈的虚拟机（Stack-based），而 Dalvik 是基于寄存器的虚拟机（Register-based）。显然，后者最大的好处在于可以根据硬件实现更大的优化，这更适合移动设备的特点。

4.1.3　应用程序（Application）

Android 的应用程序主要是用户界面（User Interface）方面的，通常用 Java 语言编写，其中还可以包含各种资源文件（放置在 res 目录中）。Java 程序和相关资源在经过编译后，会生成一个 APK 包。Android 本身提供了主屏幕（Home）、联系人（Contact）、电话（Phone）和浏览器（Browser）等众多的核心应用。同时应用程序的开发者还可以使用应用程序框架层的 API 实现自己的程序，这也是 Android 开源的巨大潜力的体现。

4.1.4　应用程序框架（Application Framework）

Android 的应用程序框架为应用程序层的开发者提供 APIs，它实际上是一个应用程序的框架。由于上层的应用程序是以 Java 构建的，因此本层次提供的首先包含了 UI 程序中所需要的各种控件，例如 Views（视图组件），其中又包括了 List（列表）、Grid（栅格）、Text Box（文本框）和 Button（按钮）等，甚至一个嵌入式的 Web 浏览器。

作为一个基本的 Android 应用程序，可以利用应用程序框架中的以下 5 个部分来构建。

- Activity（活动）。
- Broadcast Intent Receiver（广播意图接收者）。
- Service（服务）。
- Content Provider（内容提供者）。
- Intent and Intent Filter（意图和意图过滤器）。

4.2　分析 Android 应用工程文件

知识点讲解：光盘:视频\知识点\第 4 章\分析 Android 应用工程文件.avi

讲解完 Android 的整体结构之后，接下来开始讲解 Android 工程文件的组成。顶层的 Android 应用程序通常使用 Eclipse+Java 组合来实现，在 Eclipse 工程中，一个基本的 Android 项目的目录结构如图 4-2 所示。

在本节的内容中，将详细讲解 Android 应用程序工程文件中各个组成部分的具体信息。

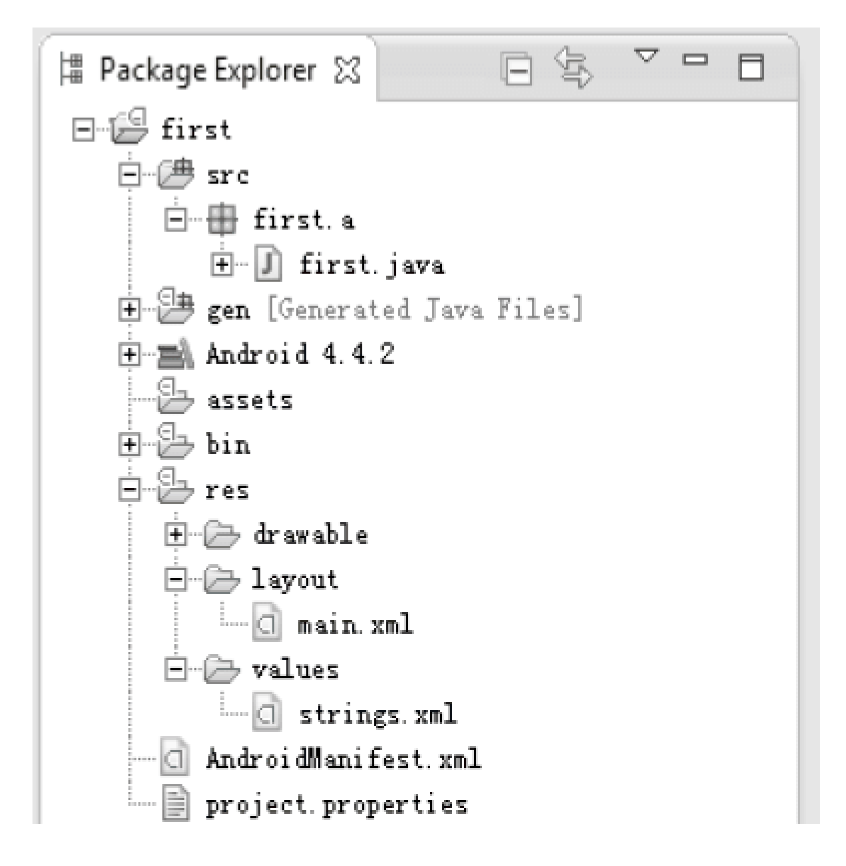
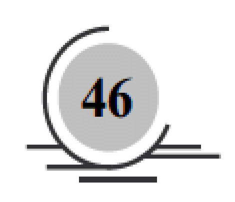


图 4-2　Android 应用工程文件组成

4.2.1　src 程序目录

src 程序目录下保存了开发人员编写的程序文件。和一般的



Java 项目一样，src 目录下保存的是项目的所有包及源文件（.java），包含了项目中的所有资源，例如，程序图标（drawable）、布局文件（layout）和常量（values）等。不同的是，在 Java 项目中没有 gen 目录，也没有每个 Android 项目都必须有的 AndroidManfest.xml 文件。

.java 格式文件是在建立项目时自动生成的，这个文件是只读模式，不能更改。R.java 文件是定义该项目所有资源的索引文件。例如下面是某项目中 R.java 文件的代码。

```
package com.yarin.Android.HelloAndroid;
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

在上述代码中定义了很多常量，并且这些常量的名字都与 res 文件夹中的文件名相同，这再次证明.java 文件中所存储的是该项目所有资源的索引。有了这个文件，在程序中使用资源将变得更加方便，可以很快地找到要使用的资源，由于这个文件不能手动编辑，所以当我们在项目中加入了新的资源时，只需要刷新一下该项目，.java 文件便自动生成了所有资源的索引。

4.2.2　设置文件 AndroidManfest.xml

文件 AndroidManfest.xml 是一个控制文件，里面包含了该项目中所使用的 Activity、Service 和 Receiver。例如下面是某项目中文件 AndroidManfest.xml 的代码。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.yarin.Android.HelloAndroid"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
       android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

在上述代码中，intent-filters 描述了 Activity 启动的位置和时间。每当一个 Activity（或者操作系统）

要执行一个操作时，它将创建出一个 Intent 的对象，这个 Intent 对象可以描述你想做什么，你想处理什么数据，数据的类型，以及一些其他信息。Android 会和每个 Application 所暴露的 intent-filter 的数据进行比较，找到最合适 Activity 来处理调用者所指定的数据和操作。下面我们来仔细分析 AndroidManfest.xml 文件，如表 4-1 所示。

表 4-1 AndroidManfest.xml 分析

参　数	说　明
manifest	根节点，描述了 package 中所有的内容
xmlns:android	包含命名空间的声明。xmlns:android=http://schemas.android.com/apk/res/android，使得 Android 中各种标准属性能在文件中使用，提供了大部分元素中的数据
package	声明应用程序包
application	包含 package 中 application 级别组件声明的根节点。此元素也可包含 application 的一些全局和默认的属性，如标签、icon、主题、必要的权限等。一个 manifest 能包含零个或一个此元素（不能大于一个）
android:icon	应用程序图标
android:label	应用程序名字
activity	activity 是与用户交互的主要工具，是用户打开一个应用程序的初始页面，大部分被使用到的其他页面也由不同的 activity 所实现，并声明在另外的 activity 标记中。注意，每一个 activity 必须有一个<activity>标记对应，无论它给外部使用或是只用于自己的 package 中。如果一个 activity 没有对应的标记，你将不能运行它。另外，为了支持运行时查找 activity，可包含一个或多个<intent-filter>元素来描述 activity 所支持的操作
android:name	应用程序默认启动的 activity
intent-filter	声明了指定的一组组件支持的 Intent 值，从而形成 Intent Filter。除了能在此元素下指定不同类型的值，属性也能放在这里来描述一个操作所需的唯一的标签、icon 和其他信息
action	组件支持的 Intent action
category	组件支持的 Intent Category。这里指定了应用程序默认启动的 activity
uses-sdk	该应用程序所使用的 sdk 版本相关

4.2.3 常量定义文件

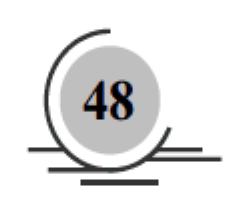
下面我们看看在资源文件中对常量的定义，例如文件 String.xml 的代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hello World, HelloAndroid!</string>
    <string name="app_name">HelloAndroid</string>
</resources>
```

上述常量定义文件的代码非常简单，只定义了两个字符串资源，请不要小看上面的几行代码。它们的内容很“露脸”，里面的字符直接显示在手机屏幕中，就像动态网站中的 HTML 一样。

4.2.4 UI 布局文件

布局（layout）文件一般位于 res\layout\main.xml 目录，通过其代码能够生成一个显示界面。例如下面的代码。



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

在上述代码中，有以下几个布局和参数。

- <LinearLayout></LinearLayout>：在这个标签中，所有元件都是按由上到下排成的。
- android:orientation：表示这个介质的版面配置方式是从上到下垂直地排列其内部的视图。
- android:layout_width：定义当前视图在屏幕上所占的宽度，fill_parent即填充整个屏幕。
- android:layout_height：定义当前视图在屏幕上所占的高度，fill_parent即填充整个屏幕。
- wrap_content：随着文字栏位的不同而改变这个视图的宽度或高度。

在上述布局代码中，使用了一个 TextView 来配置文本标签 Widget（构件），其中设置的属性 android:layout_width 为整个屏幕的宽度，android:layout_height 可以根据文字来改变高度，而 android:text 则设置了这个 TextView 要显示的文字内容，这里引用了@string 中的 hello 字符串，即 String.xml 文件中的 hello 所代表的字符串资源。hello 字符串的内容"Hello World, HelloAndroid!"这就是我们在 HelloAndroid 项目运行时看到的字符串。

注意： 上面介绍的文件只是主要文件，在项目中需要我们自行编写。在项目中还有很多其他的文件，那些文件很少需要我们编写，所以在此就不进行讲解了。

4.3　5 大核心组件

知识点讲解：光盘:视频\知识点\第 4 章\5 大核心组件.avi

一个典型的 Android 应用程序通常由 5 个组件组成，这 5 个组件实现了 Android 应用程序的核心功能。本节将一一讲解这 5 大组件的基本知识，为学习本书后面的知识打下基础。

4.3.1　Activity 界面组件

Activities 是这 5 个组件中最常用的一个组件。程序中 Activity 通常的表现形式是一个单独的界面（screen）。每个 Activity 都是一个单独的类，它扩展实现了 Activity 基础类。这个类显示为一个由 Views 组成的用户界面，并响应事件。大多数程序有多个 Activity。例如，一个文本信息程序有这么几个界面：显示联系人列表界面、写信息界面、查看信息界面或者设置界面等。每个界面都是一个 Activity，切换到另一个界面就是载入一个新的 Activity。某些情况下，一个 Activity 可能会给前一个 Activity 返回值——如一个让用户选择相片的 Activity 会把选择到的相片返回给其调用者。

打开一个新界面后，前一个界面就被暂停，并放入历史栈中（界面切换历史栈）。使用者可以回溯前面已经打开的存放在历史栈中的界面，也可以从历史栈中删除没有界面价值的界面。Android 在历史栈中保留程序运行产生的所有界面：从第一个界面，到最后一个。

4.3.2 Intent 切换组件

Android 通过一个专门的 Intent 类来进行界面的切换。Intent 描述了程序想做什么（Intent 意为意图、目的、意向）。Intent 类还有一个相关类 IntentFilter。Intent 是一个请求来做什么事情，IntentFilter 则描述了一个 Activity（或下文的 IntentReceiver）能处理什么意图。显示某人联系信息的 Activity 使用了一个 IntentFilter，就是说它知道如何处理应用到此人数据的 VIEW 操作。Activities 在文件 AndroidManifest.xml 中使用 IntentFilters。

通过解析 Intents 可以实现 Activity 的切换，我们可以使用 startActivity(myIntent)启用新的 Activity。系统会考察所有安装程序的 IntentFilters，然后找到与 myIntent 匹配最好的 IntentFilters 所对应的 Activity。这个新 Activity 能够接收 Intent 传来的消息，并因此被启用。解析 Intents 的过程发生在 startActivity 被实时调用时，这样做有如下两个好处。

（1）Activities 仅发出一个 Intent 请求，便能重用其他组件的功能。

（2）Activities 可以随时被替换为有等价 IntentFilter 的新 Activity。

4.3.3 Service 服务组件

Service 是一个没有 UI 且长驻系统的代码，最常见的例子是媒体播放器从播放列表中播放歌曲。在媒体播放器程序中，可能有一个或多个 Activity 让用户选择播放的歌曲。然而在后台播放歌曲时无须 Activity 干涉，因为用户希望在音乐播放的同时能够切换到其他界面。既然这样，媒体播放器 Activity 需要通过 Context.startService()启动一个 Service，这个 Service 在后台运行以保持继续播放音乐。在媒体播放器被关闭之前，系统会保持音乐后台播放 Service 的正常运行。可以用 Context.bindService()方法连接到一个 Service 上（如果 Service 未运行的话，连接后还会启动它），连接后就可以通过一个 Service 提供的接口与 Service 进行通话。对音乐 Service 来说，提供了暂停和重放等功能。

1. 如何使用服务

在 Android 系统中有如下两种使用服务的方法。

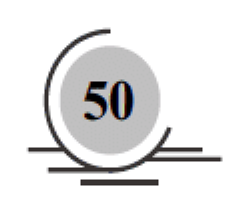
（1）通过调用 Context.startService()启动服务，调用 Context.stopService()结束服务，startService()可以传递参数给 Service。

（2）通过调用 Context.bindService()启动，调用 Context.unbindService()结束，还可以通过 ServiceConnection 访问 Service。二者可以混合使用，比如说可以先 startService()再 unbindService()。

2. Service 的生命周期

在 startService()后，即使调用 startService()的进程结束了，Service 还仍然存在，一直到有进程调用 stopService()或者 Service 自己灭亡（stopSelf()）为止。

在 bindService()后，Service 就和调用 bindService()的进程同生共死，也就是说当调用 bindService()的进程死了，那么它绑定的 Service 也要跟着被结束，当然期间也可以调用 unbindService()让 Service



结束。

当混合使用上述两种方式时，例如你 startService()了，我 bindService()了，那么只有你 stopService()而且我也 unbindService()了，这个 Service 才会被结束。

3. 进程生命周期

Android 系统将会尝试保留那些启动了的或者绑定了的服务进程，具体说明如下所示。

（1）如果该服务正在进程的 onCreate()、onStart()或者 onDestroy()这些方法中执行，那么主进程将会成为一个前台进程，以确保此代码不会被停止。

（2）如果服务已经开始，那么它的主进程的重要性会低于所有的可见进程，但是会高于不可见进程。由于只有少数几个进程是用户可见的，所以只要不是内存特别低，该服务就不会停止。

（3）如果有多个客户端绑定了服务，只要客户端中的一个对于用户是可见的，就可以认为该服务可见。

4.3.4　Broadcast/Receiver 广播机制组件

当要执行一些与外部事件相关的代码时，比如来电响铃时或者半夜时就可能用到 IntentReceiver。尽管 IntentReceiver 使用 NotificationManager 来通知用户一些好玩的事情发生，但是没有 UI。IntentReceiver 可以在文件 AndroidManifest.xml 中声明，也可以使用 Context.registerReceiver()来声明。当一个 IntentReceiver 被触发时，如果需要，系统自然会自动启动程序。程序也可以通过 Context.broadcastIntent()来发送自己的 Intent 广播给其他程序。

4.3.5　ContentProvider 存储组件

应用程序把数据存放在一个 SQLite 数据库格式文件里，或者存放在其他有效设备里。如果想让其他程序能够使用我们程序中的数据，此时 ContentProvider 就很有用了。ContentProvider 是一个实现了一系列标准方法的类，这个类使得其他程序能存储、读取某种 ContentProvider 可处理的数据。

4.4　进程和线程

知识点讲解：光盘:视频\知识点\第 4 章\进程和线程.avi

进程和线程很容易理解，在 PC 机的 Windows 系统中都有一个进程管理器，当打开后，会显示当前运行的所有程序。同样在 Android 系统中也有进程，当某个组件第一次运行的时候，Android 会启动一个进程。在默认情况下，所有的组件和程序运行在这个进程和线程中，也可以安排组件在其他的进程或者线程中运行。本节将详细讲解 Android 系统中进程和线程的基本知识。

4.4.1　应用程序的生命周期

自然界的事物都有自己的生命周期，例如人的生、老、病、死。作为一个 Android 应用程序也如同自然界的生物一样，有自己的生命周期。我们开发一个程序的目的是为了完成一个功能，例如银行计算加息的软件，每当一个用户去柜台办理取款业务时，银行工作人员便启动了我们这个程序的生命，

当用这个软件完成利息计算时，这个软件当前的任务就完成了，此时就需要结束自己的使命。肯定有人提出疑问：生生死死多么麻烦，就让这个程序一直是“活着”的状态，一个用户办理完取款业务后，继续等着下一个用户办理取款业务，这样这个程序就“长生不老”了。其实谁都想自己的程序“长生不老”，但是很不幸，我们不能这样做。原因是计算机的处理性能是一定的，一个人、两个人、三个人计算机可以处理这个任务，但是一个安装这个软件的机器一天会处理上千上万个取款业务，如果它们都一直活着，一台有限配置的计算机能承受得了吗？

由此可见，应用程序的生命周期就是一个程序的存活时间，即在什么时间内有效。Android 是一个构建在 Linux 之上的开源移动开发平台，在 Android 中，多数情况下每个程序都是在各自独立的 Linux 进程中运行的。当一个程序或其某些部分被请求时，它的进程就“出生”了；当这个程序没有必要再运行下去且系统需要回收这个进程的内存用于其他程序时，这个进程就“死亡”了。可以看出，Android 程序的生命周期是由系统控制而非程序自身直接控制。这和我们编写桌面应用程序时的思维有一些不同，一个桌面应用程序的进程也是在其他进程或用户请求时被创建，但是往往是在程序自身收到关闭请求后执行一个特定的动作（比如从 main()函数中返回）而导致进程结束的。要想做好某种类型的程序或者某种平台下的程序的开发，最关键的就是要弄清楚这种类型的程序或整个平台下的程序的一般工作模式并熟记在心。在 Android 中，程序的生命周期控制就是属于这个范畴。

开发者必须理解不同的应用程序组件，尤其是 Activity、Service 和 Intent Receiver，需要了解这些组件是如何影响应用程序生命周期的。如果不正确地使用这些组件，可能会导致系统终止正在执行重要任务的应用程序进程。

一个常见的进程生命周期漏洞的例子是 Intent Receiver（意图接收器），当 Intent Receiver 在 onReceive()方法中接收到一个 Intent（意图）时，它会启动一个线程，然后返回。一旦返回，系统将认为 Intent Receiver 不再处于活动状态，因而 Intent Receiver 所在的进程也就不再有用了（除非该进程中还有其他的组件处于活动状态）。因此，系统可能会在任意时刻终止该进程以回收占有的内存。这样进程中创建出的那个线程也将被终止。解决这个问题的方法是从 Intent Receiver 中启动一个服务，让系统知道进程中还有处于活动状态的工作。为了使系统能够正确决定在内存不足时应该终止哪个进程，Android 根据每个进程中运行的组件及组件的状态把进程放入一个 Importance Hierarchy（重要性分级）中。

4.4.2 什么是进程

组件运行的进程是由 manifest file 控制的。组件的节点一般都包含一个 process 属性，例如<activity>、<service>、<receiver>和<provider>节点。属性 process 可以设置组件运行的进程，可以配置组件在一个独立进程中运行，或者多个组件在同一个进程中运行，甚至可以多个程序在一个进程中运行，当然前提是这些程序共享一个 User ID 并给定同样的权限。另外<application>节点也包含了 process 属性，用来设置程序中所有组件的默认进程。

当更加常用的进程无法获取足够内存时，Android 会智能地关闭不常用的进程，当下次启动程序时会重新启动这些进程。当决定哪个进程需要被关闭的时候，Android 会考虑哪个对用户更加有用。例如 Android 会倾向于关闭一个长期不显示在界面的进程来支持一个经常显示在界面的进程。是否关闭一个进程决定于组件在进程中的状态。

进程的类型多种多样，按照重要的程度主要包括如下几类进程。

（1）前台进程（Foreground）

前台进程是看得见的，与用户当前正在做的事情密切相关，不同的应用程序组件能够通过不同的方法将它的宿主进程移到前台。在如下的任何一个条件下系统会把进程移动到前台。

- 进程正在屏幕的最前端运行一个与用户交互的活动（Activity），它的onResume()方法被调用。
- 进程有一个正在运行的Intent Receiver（它的IntentReceiver.onReceive方法正在执行）。
- 进程有一个服务（Service），并且在服务的某个回调函数（Service.onCreate、Service.onStart或Service.onDestroy）内有正在执行的代码。

（2）可见进程（Visible）

可见进程也是可见的，它有一个可以被用户从屏幕上看到的活动，但不在前台（它的 onPause()方法被调用）。假如前台的活动是一个对话框，以前的活动隐藏在对话框之后就会出现这种进程。可见进程非常重要，一般不允许被终止，除非是为了保证前台进程的运行而不得不终止它。

（3）服务进程（Service）

服务进程是无法看见的，拥有一个已经用 startService()方法启动的服务。虽然用户无法直接看到这些进程，但它们做的事情却是用户所关心的（如后台 MP3 回放或后台网络数据的上传下载）。所以系统将一直运行这些进程，除非内存不足以维持所有的前台进程和可见进程。

（4）后台进程（Background）

后台进程也是看不见的，只有打开之后才能看见。例如迅雷下载，我们可以将其最小化，虽然在桌面上看不见了，但是它一直在进行下载的工作，拥有一个当前用户看不到的活动（它的 onStop()方法被调用）。这些进程对用户体验没有直接的影响。如果它们正确执行了活动生命周期，系统可以在任意时刻终止该进程以回收内存，并提供给前面 3 种类型的进程使用。系统中通常有很多这样的进程在运行，因此要将这些进程保存在 LRU 列表中，以确保当内存不足时用户最近看到的进程最后一个被终止。

（5）空进程（Empty）

空进程是指不拥有任何活动的应用程序组件的进程。保留这种进程的唯一原因是在下次应用程序的某个组件需要运行时，不需要重新创建进程，这样可以提高启动速度。系统将以进程中当前处于活动状态组件的重要程度为基础对进程进行分类。进程的优先级可能也会根据该进程与其他进程的依赖关系而增长。假如进程 A 通过在进程 B 中设置 Context.BIND_AUTO_CREATE 标记或使用 ContentProvider 被绑定到一个服务（Service），那么进程 B 在分类时至少要被看成与进程 A 同等重要。

例如 Activity 的状态转换图如图 4-3 所示。

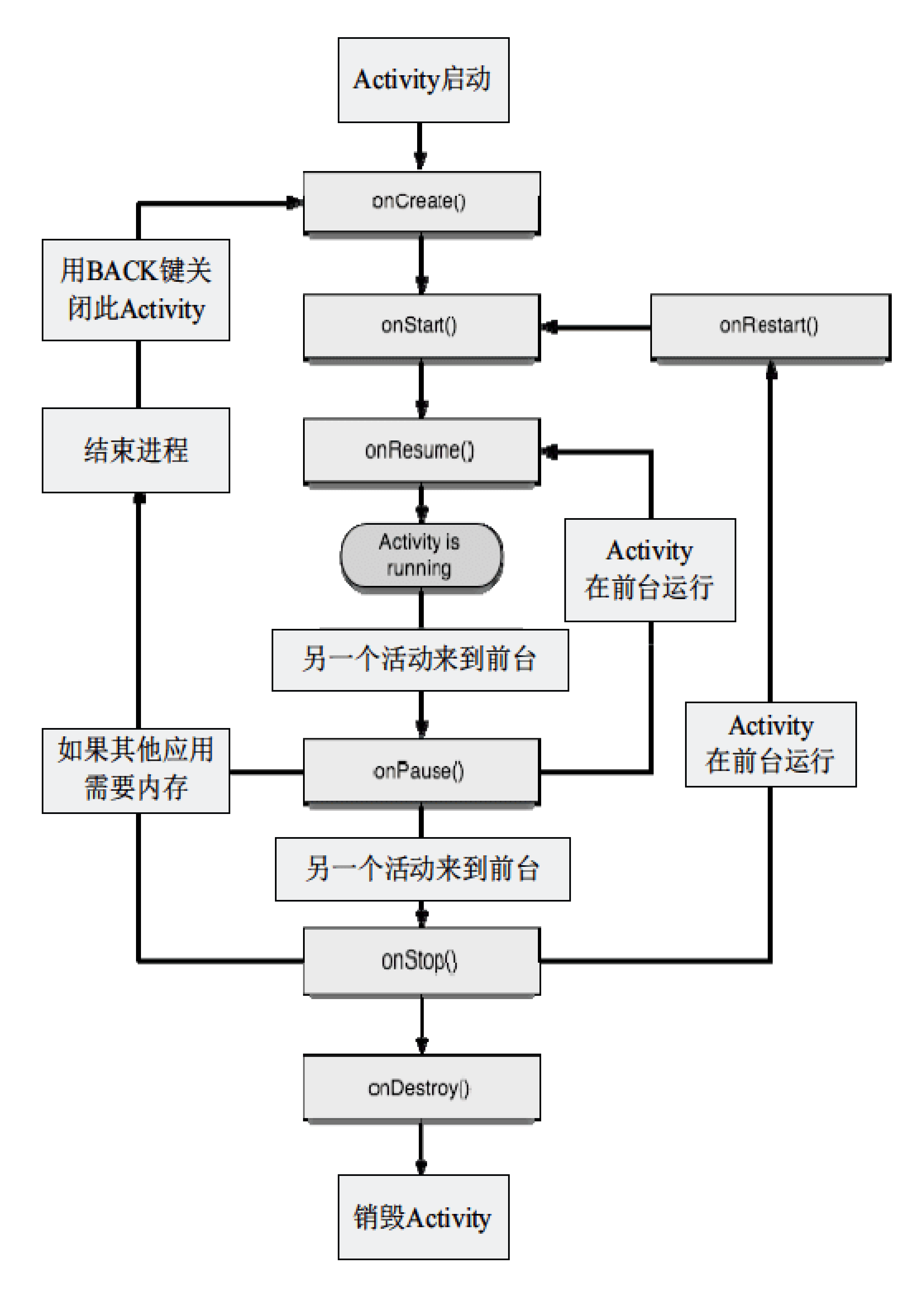


图 4-3　Activity 状态转换图

图 4-3 所示的状态的变化是由 Android 内存管理器决定的，Android 会首先关闭那些包含 Inactive Activity 的应用程序，然后关闭 Stopped 状态的程序。只有在极端情况下才会移除 Paused 状态的程序。

4.4.3 什么是线程

当用户界面需要很快对用户进行响应时，就需要将一些费时的操作，如网络连接、下载或者非常占用服务器时间的操作等放到其他线程。也就是说，即使为组件分配了不同的进程，有时也需要再分配线程。

线程是通过 Java 的标准对象 Thread 来创建的，在 Android 中提供了如下方便的管理线程的方法。

- ❑ Looper在线程中运行一个消息循环。
- ❑ Handler 传递一个消息。
- ❑ HandlerThread创建一个带有消息循环的线程。
- ❑ Android让一个应用程序在单独的线程中，指导它创建自己的线程。
- ❑ 应用程序组件（Activity、Service、Broadcast/Receiver）所有都在理想的主线程中实例化。
- ❑ 没有一个组件应该执行长时间或是阻塞操作（例如网络呼叫或是计算循环），当被系统调用时，这将中断所有在该进程的其他组件。
- ❑ 可以创建一个新的线程来执行长期操作。

4.5 Android 和 Linux 的关系

知识点讲解：光盘:视频\知识点\第 4 章\Android 和 Linux 的关系.avi

虽然 Android 系统采用 Linux 作为其内核，但是谷歌对 Linux 内核做了修改，以适应其在 Android 移动设备上的应用。Android 开始是作为 Linux 的一个分支，后来由于无法并入 Linux 的主开发树，曾经被 Linux 内核组从开发树中删除。直到 2012 年 5 月 18 日，Linux Kernel 3.3 发布后又被加入到 Linux 的主开发树中。本节将详细讲解 Android 系统和 Linux 系统之间的关系。

4.5.1 Android 继承于 Linux

Android 是在 Linux 的内核基础之上运行的，提供的核心系统服务包括安全、内存管理、进程管理、网络组和驱动模型等内容。内核部分还相当于一个介于硬件层和系统中其他软件组之间的一个抽象层次。但是严格来说它不算是 Linux 操作系统。

因为 Android 内核是由标准的 Linux 内核修改而来的，所以继承了 Linux 内核的诸多优点，保留了 Linux 内核的主题架构。同时 Android 按照移动设备的需求，在文件系统、内存管理、进程间通信机制和电源管理方面进行了修改，添加了相关的驱动程序和必要的新功能。但是和其他精简的 Linux 系统相比（比如 uClinux），Android 很大程度地保留了 Linux 的基本架构，因此 Android 的应用性和扩展性更强。当前 Android 版本对应的 Linux 内核版本如下。

- ❑ Android 1.5：Linux-2.6.27。
- ❑ Android 1.6：Linux-2.6.29。
- ❑ Android 2.0,2.1：Linux-2.6.29。
- ❑ Android 2.2：Linux-2.6.32.9。
- ❑ Android 4.3：Linux-3.4。

4.5.2　Android 和 Linux 内核的区别

Android 系统的系统层面的底层是 Linux，中间加上了一个叫做 Dalvik 的 Java 虚拟机，表面层上面是 Android 运行库。每个 Android 应用都运行在自己的进程上，享有 Dalvik 虚拟机为它分配的专有实例。为了支持多个虚拟机在同一个设备上高效运行，Dalvik 被改写过。

Dalvik 虚拟机执行的是 Dalvik 格式的可执行文件（.dex）——该格式经过优化，以降低内存耗用到最低。Java 编译器将 Java 源文件转为 class 文件，class 文件又被内置的 dx 工具转化为 dex 格式文件，这种文件在 Dalvik 虚拟机上注册并运行。

Android 系统的应用软件都是运行在 Dalvik 之上的 Java 软件，而 Dalvik 是运行在 Linux 中的，在一些底层功能——比如线程和低内存管理方面，Dalvik 虚拟机是依赖 Linux 内核的。由此可见，可以说 Android 是运行在 Linux 之上的操作系统，但是它本身不能算是 Linux 的某个版本。

Android 内核和 Linux 内核的差别主要体现在 11 个方面，接下来将一一简要介绍。

（1）Android Binder

其源代码位于：

```
drivers/staging/android/binder.c
```

Android Binder 是基于 OpenBinder 框架的一个驱动，用于提供 Android 平台的进程间通信（Inter-Process Communication，IPC）。原来的 Linux 系统上层应用的进程间通信主要是 D-bus（Desktop Bus），采用消息总线的方式来进行 IPC。

（2）Android 电源管理（PM）

Android 电源管理是一个基于标准 Linux 电源管理系统的轻量级的 Android 电源管理驱动，针对嵌入式设备做了很多优化。利用锁和定时器来切换系统状态，控制设备在不同状态下的功耗，以达到节能的目的。

Android 电源管理的源代码分别位于：

```
kernel/power/earlysuspend.c
kernel/power/consoleearlysuspend.c
kernel/power/fbearlysuspend.c
kernel/power/wakelock.c
kernel/power/userwakelock.c
```

（3）低内存管理器（Low Memory Killer）

Android 中的低内存管理器和 Linux 标准的 OOM（Out Of Memory）相比，其机制更加灵活，它可以根据需要杀死进程来释放需要的内存。低内存管理器的代码很简单，关键的一个函数是 Lowmem_shrinker。作为一个模块在初始化时调用 register_shrinker 注册了一个 Lowmem_shrinker，它会被 VM 在内存紧张的情况下调用。Lowmem_shrinker 完成具体操作。简单说就是寻找一个最合适的进程杀死，从而释放它占用的内存。

低内存管理器的源代码位于：drivers/staging/android/lowmemorykiller.c。

（4）匿名共享内存（Ashmem）

匿名共享内存为进程间提供大块共享内存，同时为内核提供回收和管理这个内存的机制。如果一个程序尝试访问 Kernel 释放的一个共享内存块，它将会收到一个错误提示，然后重新分配内存并重载数据。

匿名共享内存的源代码位于：mm/ashmem.c。

（5）Android PMEM（Physical）

PMEM 用于向用户空间提供连续的物理内存区域，DSP 和某些设备只能工作在连续的物理内存上。驱动中提供了 mmap、open、release 和 ioctl 等接口。

Android PMEM 的源代码位于：drivers/misc/pmem.c。

（6）Android Logger

Android Logger 是一个轻量级的日志设备，用于抓取 Android 系统的各种日志，是 Linux 所没有的。其源代码位于：drivers/staging/android/logger.c。

（7）Android Alarm

Android Alarm 提供了一个定时器用于把设备从睡眠状态唤醒，同时它也提供了一个即使在设备睡眠时也会运行的时钟基准。

Android Alarm 的源代码位于：

- drivers/rtc/alarm.c
- drivers/rtc/alarm-dev.c

（8）USB Gadget 驱动

USB Gadget 驱动是一个基于标准 Linux USB Gadget 驱动框架的设备驱动，Android 的 USB 驱动是基于 Gadget 框架的。

USB Gadget 驱动的源代码位于：

- drivers/usb/gadget/android.c
- drivers/usb/gadget/f_adb.c
- drivers/usb/gadget/f_mass_storage.c

（9）Android Ram Console

为了提供调试功能，Android 允许将调试日志信息写入一个被称为 RAM Console 的设备里，它是一个基于 RAM 的 Buffer。

Android Ram Console 的源代码位于：drivers/staging/android/ram_console.c。

（10）Android timed device

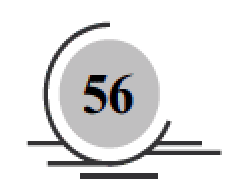
Android timed device 提供了对设备进行定时控制功能，目前仅仅支持 vibrator 和 LED 设备。其源代码位于：drivers/staging/android/timed_output.c(timed_gpio.c)。

（11）Yaffs2 文件系统

在 Android 系统中，采用 Yaffs2 作为 MTD NAND Flash 文件系统。Yaffs2 是一个快速稳定的应用于 NAND 和 NOR Flash 的跨平台的嵌入式设备文件系统，同其他 Flash 文件系统相比，Yaffs2 使用更小的内存来保存它的运行状态，因此它占用内存小；Yaffs2 的垃圾回收非常简单而且快速，因此能达到更好的性能；Yaffs2 在大容量的 NAND Flash 上性能表现尤为明显，非常适合大容量的 Flash 存储。

Yaffs2 文件系统源代码位于 fs/yaffs2/目录下。

Android 是在 Linux 的内核基础之上运行的，提供的核心系统服务包括安全、内存管理、进程管理、网络组和驱动模型等内容。内核部分还相当于一个介于硬件层和系统中其他软件组之间的一个抽象层次。但是严格来说它不算是 Linux 操作系统。



4.6　编写第一段 Android 程序

知识点讲解：光盘:视频\知识点\第 4 章\编写第一段 Android 程序.avi

经过本书前面内容的学习，已经了解 Android 系统诞生和具体架构知识，也了解了搭建 Android 开发环境的方法。在本节的内容中，将创建一个 Android 应用程序项目，演示开发 Android 应用程序的具体流程。本实例的功能是在手机屏幕中显示问候语“你好我的朋友！”，在具体开始之前先做一个简单的流程规划，如图 4-4 所示。

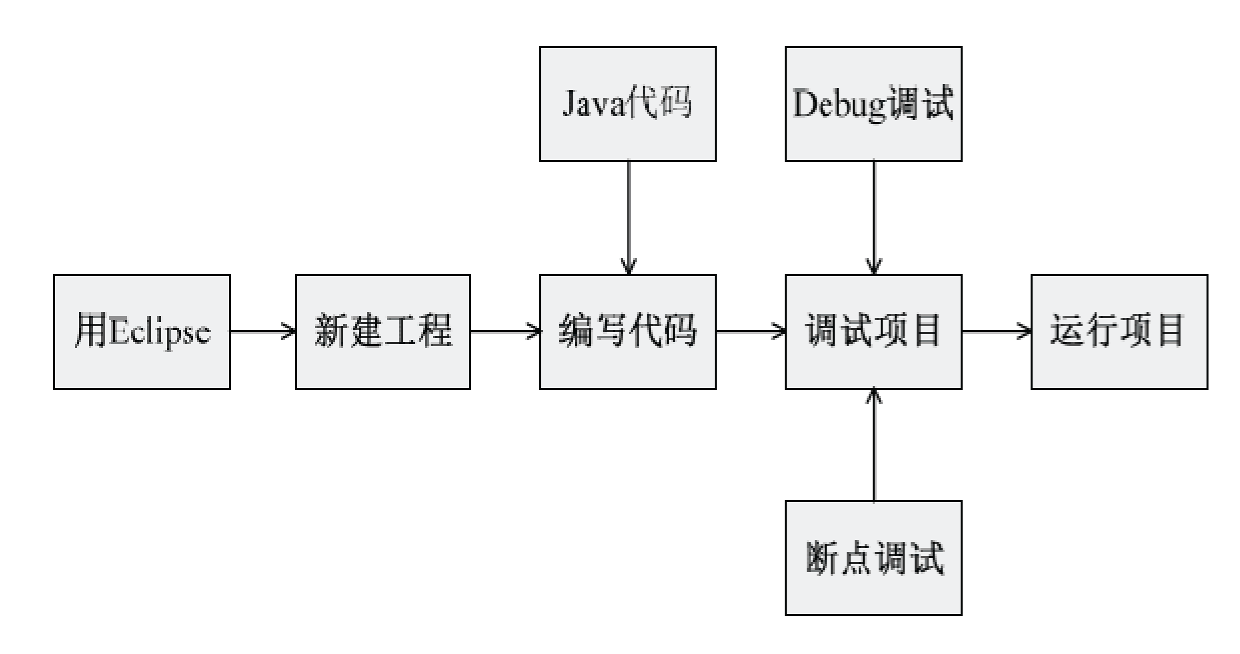


图 4-4　规划流程图

实例	功能	源码路径
实例 4-1	演示第一个 Android 应用程序	光盘:\daima\4\first

4.6.1　新建一个 Android 工程

1. 新建工程

（1）在 Eclipse 中依次选择 File | New | Project 命令新建一个工程文件，如图 4-5 所示。

（2）选择 Android Project 选项，单击 Next 按钮。

（3）在弹出的 New Android Project 对话框中设置工程信息，如图 4-6 所示。

在图 4-6 所示的界面中依次设置工程名字、包名字、Activity 名字和应用名字。

2. 编写代码和代码分析

现在已经创建了一个名为 first 的工程文件，打开文件 first.java，会显示自动生成的如下代码。

```
package first.a;
import android.app.Activity;
import android.os.Bundle;
public class fistMM extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

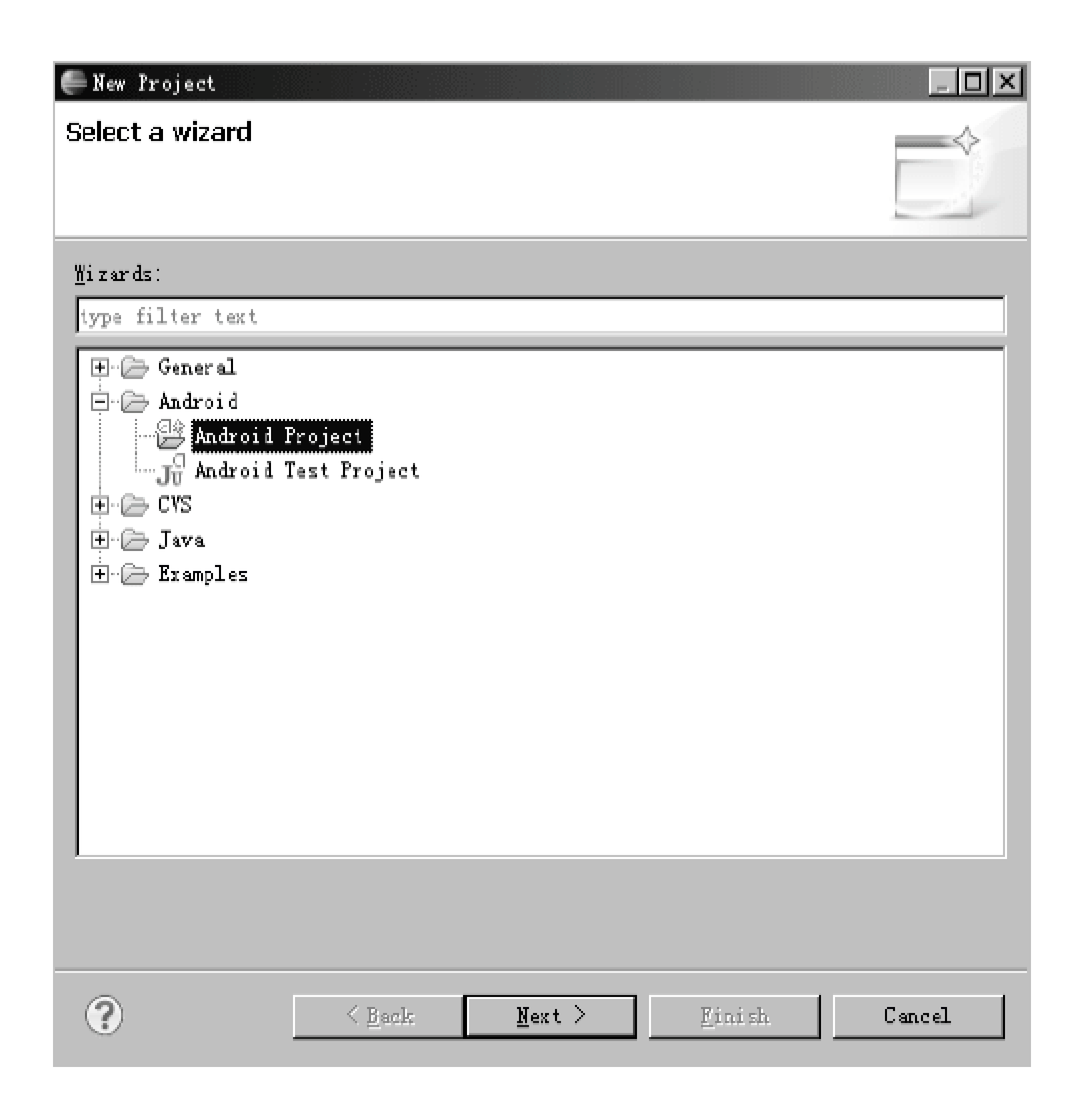




图 4-5　新建工程文件

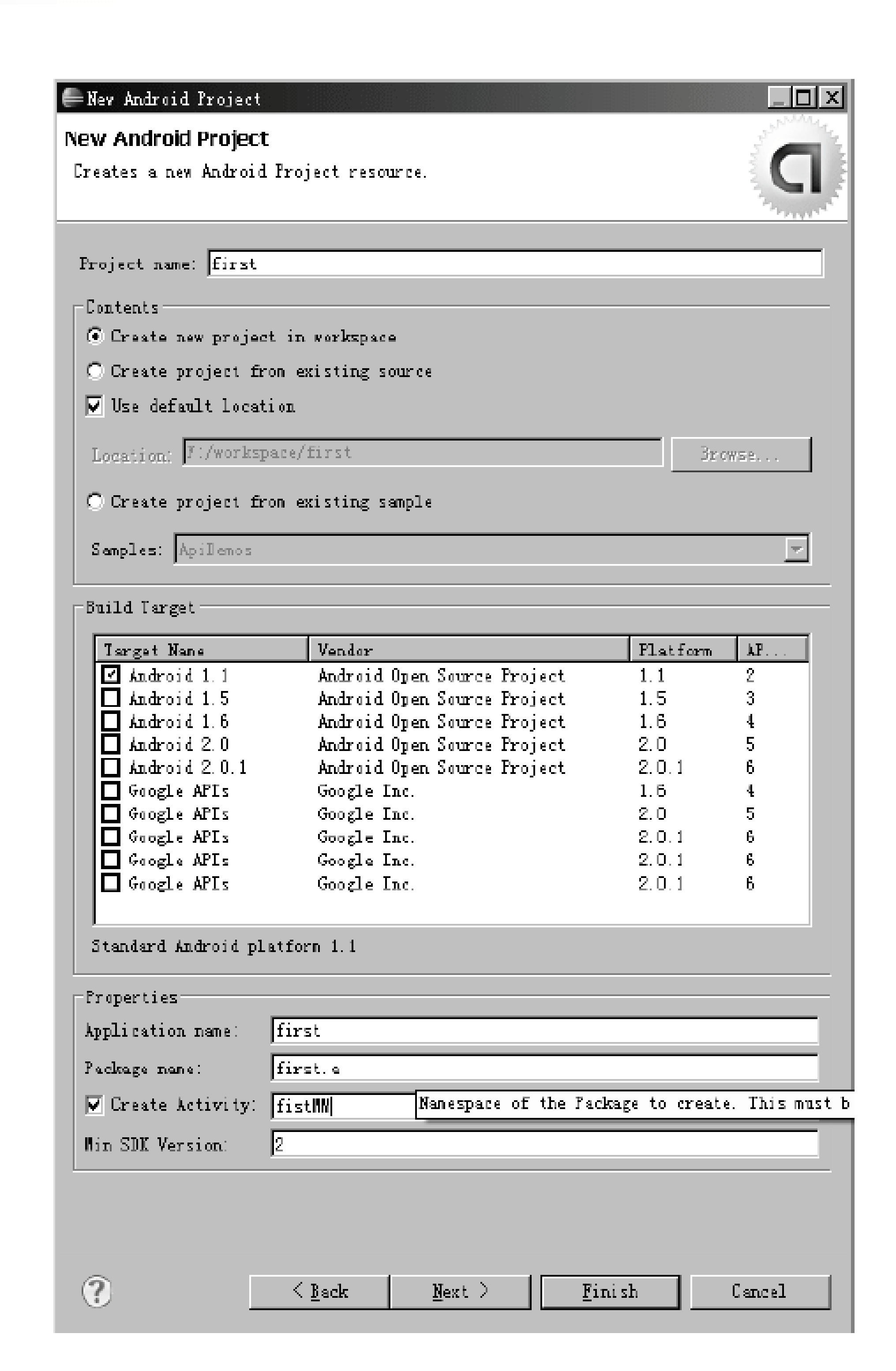


图 4-6　设置工程

如果此时运行程序，将不会显示任何内容。此时我们可以对上述代码进行修改，让程序输出“你好我的朋友!”。具体代码如下所示。

```
package first.a;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class fistMM extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv = new TextView(this);
        tv.setText("你好我的朋友！ ");
        setContentView(tv);
    }
}
```

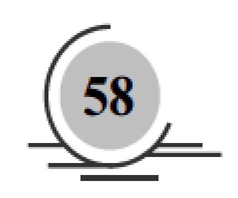
经过上述代码改写后，应该可以在屏幕中输出“你好我的朋友!”，完全符合预期的要求。

4.6.2　调试程序

Android 调试一般分为 3 个步骤，分别是设置断点、Debug 调试和断点调试。

（1）设置断点

此处的设置断点和 Java 中的方法一样，可以通过双击代码左边的区域进行断点设置，如图 4-7 所示。



为了调试方便，可以设置显示代码的行数。只需在代码左侧的空白部分右键单击，在弹出的快捷菜单中选择 Show Line Numbers，如图 4-8 所示。

（2）Debug 调试

Debug Android 调试项目的方法和普通 Debug Java 调试项目的方法类似，唯一的不同是在调试项目时选择 Android Application 命令。具体方法是右键单击项目名，在弹出的快捷菜单中选择 Debug As | Android Application 命令，如图 4-9 所示。

first Manifest　first.java

```
package first.a;
import android.app.Activity;

public class first extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv = new TextView(this);
        tv.setText("你好我的朋友! ");
        setContentView(tv);

    }
}
```

图 4-7　设置断点

Toggle Breakpoint
Toggle Enablement
Open Super Implementation　Ctrl+1
Add Bookmark...
Add Task...
Show Quick Diff　Ctrl+Shift+Q
Show Line Numbers
Folding
Preferences...
Breakpoint Properties...

图 4-8　显示行数

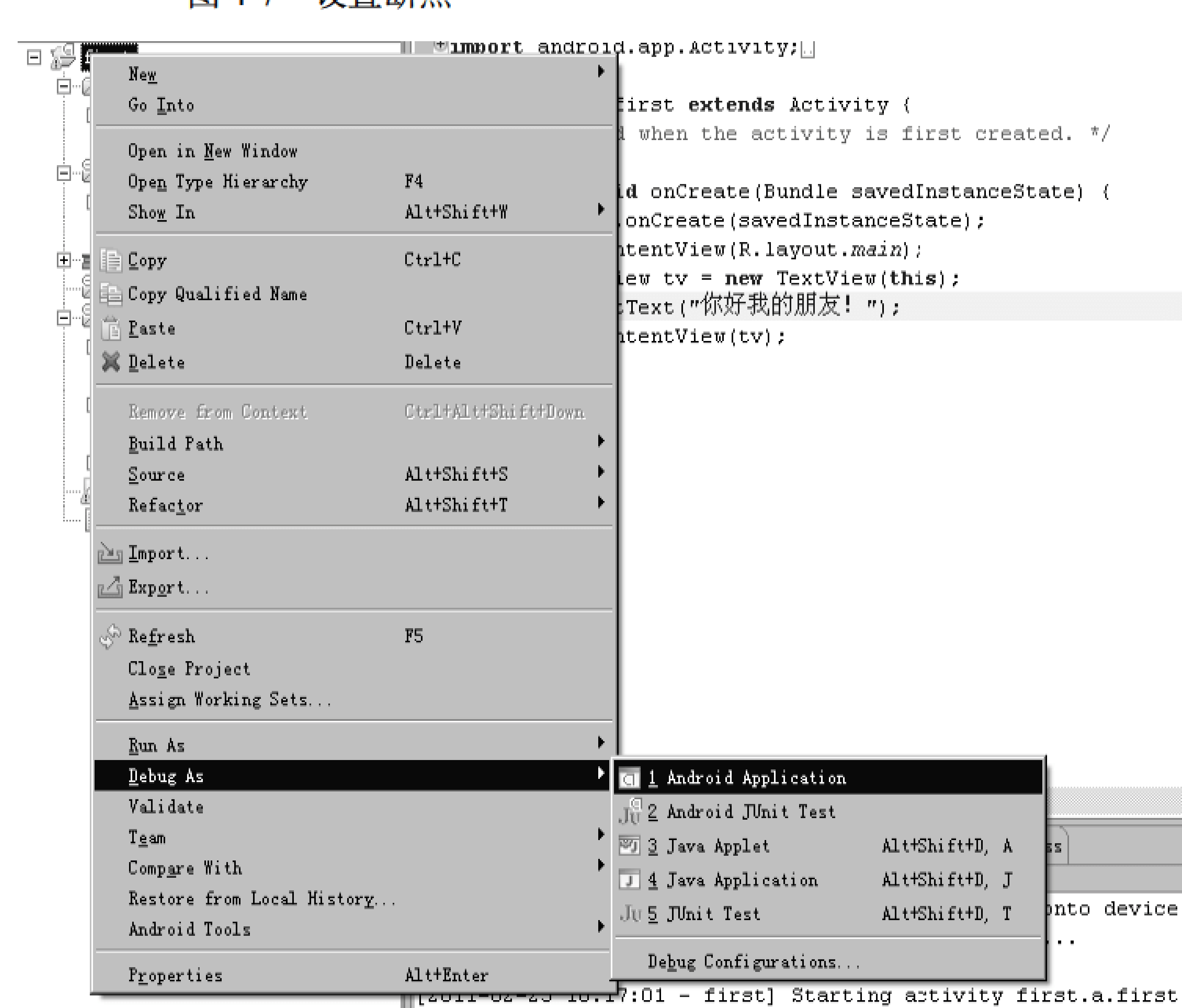


图 4-9　Debug 项目

（3）断点调试

可以进行单步调试，具体调试方法和调试普通 Java 程序的方法类似，调试界面如图 4-10 所示。

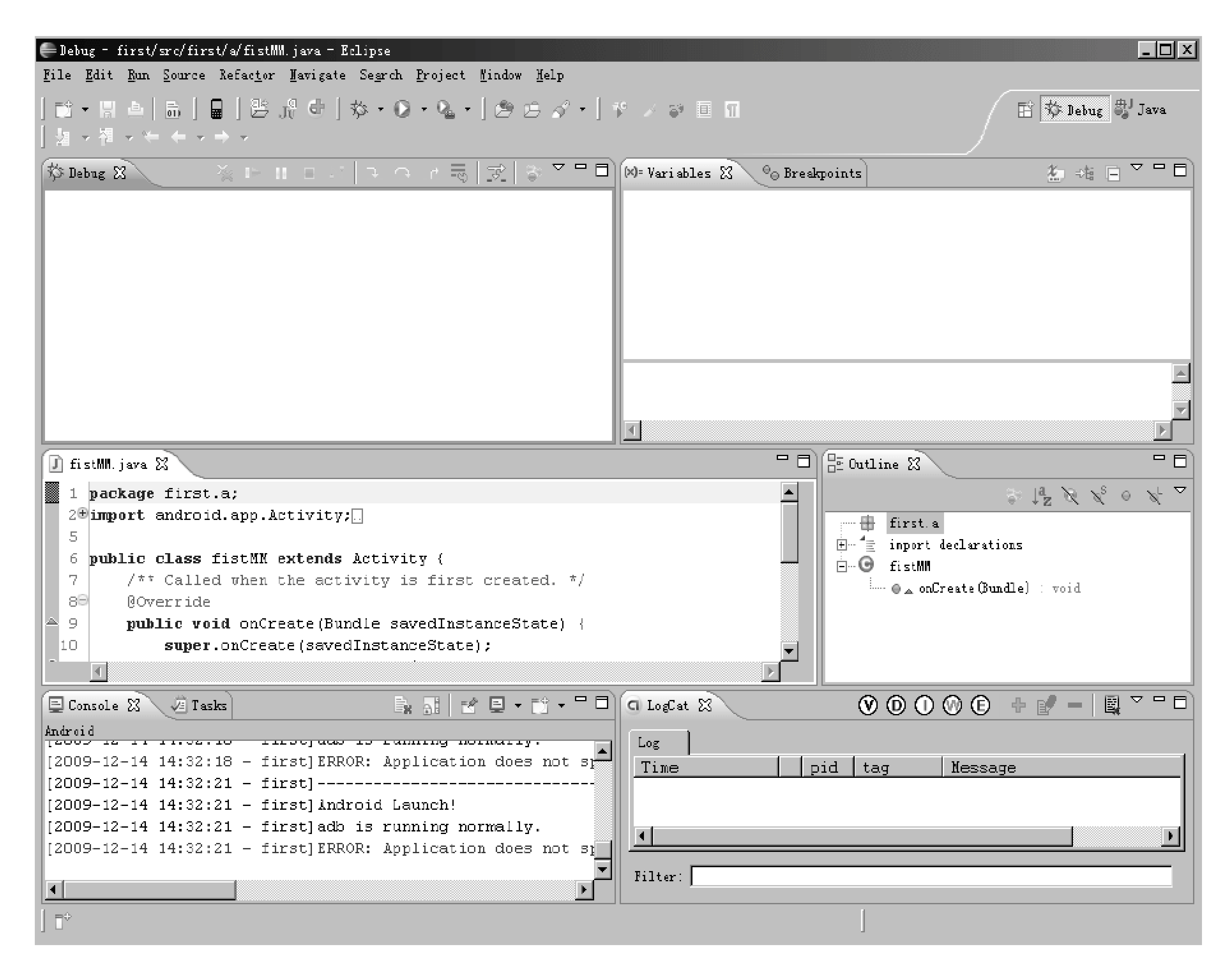


图 4-10　调试界面

4.6.3　运行程序

将上述代码保存后就可运行这段程序了，具体过程如下。

（1）右键单击项目名，在弹出的快捷菜单中依次选择 Run As | Android Application 命令，如图 4-11 所示。

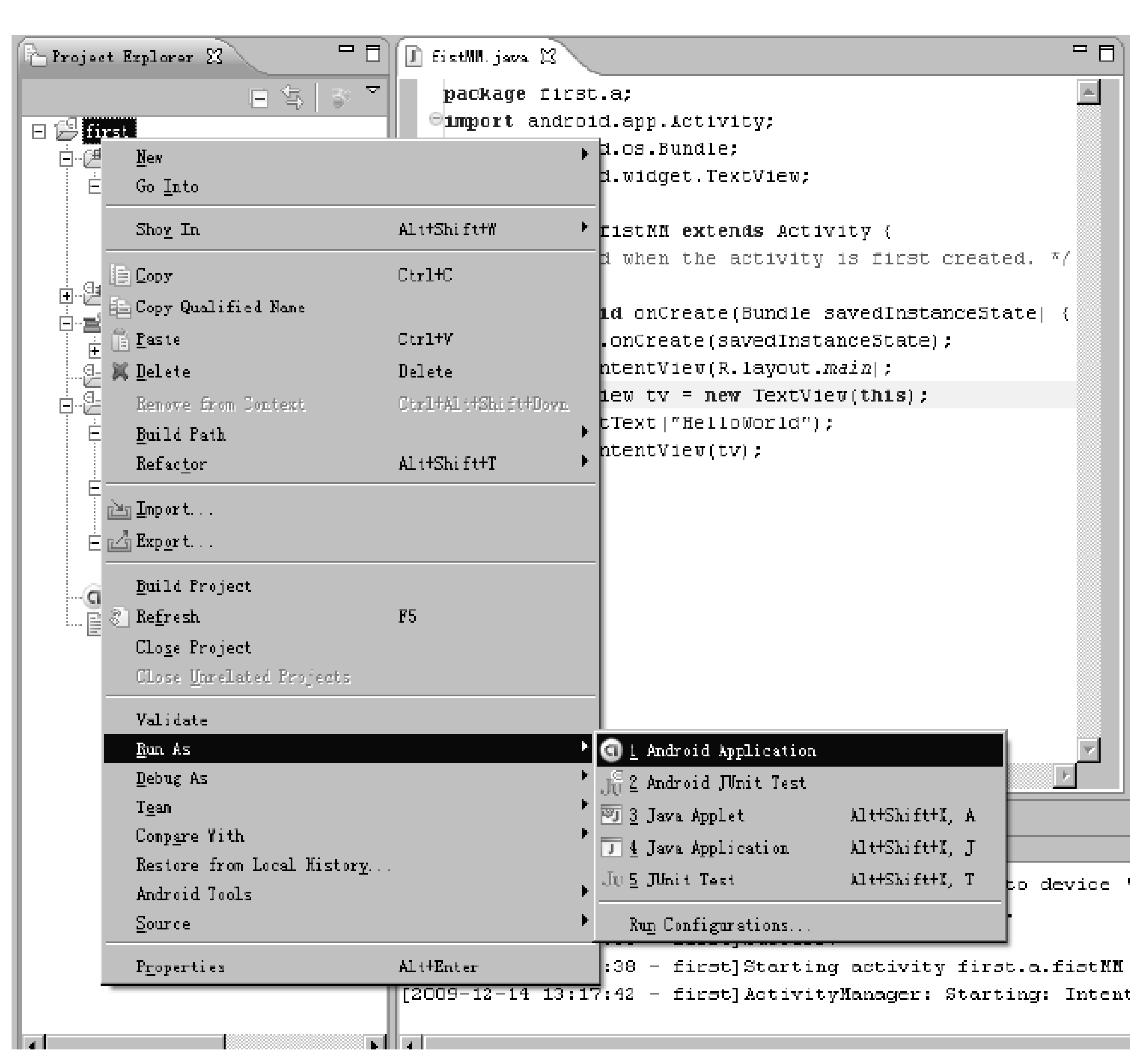
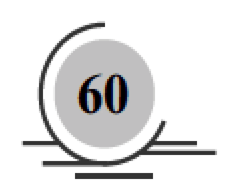


图 4-11　开始调试



（2）此时工程开始运行，运行完成后在屏幕中输出“你好我的朋友！”这段文字，如图 4-12 所示。



图 4-12　运行结果

这样，我们的 Android 应用程序便创建并调试运行完毕了。由此可见，通过 Eclipse 工具可以高效地开发出我们需要的 Android 应用程序。

第 2 篇　系统分析篇



第 5 章　Android 传感器系统架构详解

传感器是近年来随着物联网这一概念的流行而推出的，现在人们已经逐渐地了解了传感器这一概念。其实传感器在大家的日常生活中经常见到甚至是用到，比如楼宇的声控楼梯灯和马路上的路灯等。在开发 Android 外设项目程序时，经常通过传感器来建立 Android 软件系统和外部硬件设备之间的纽带。本章将详细讲解 Android 系统中传感器系统的基本架构知识，为读者学习本书后面的知识打下基础。

5.1　Android 传感器系统概述

知识点讲解：光盘:视频\知识点\第 5 章\Android 传感器系统概述.avi

在 Android 系统中提供的主要传感器有：加速度传感器、磁场、方向、陀螺仪、光线、压力、温度和接近等。传感器系统会主动对上层报告传感器精度和数据的变化，并且提供了设置传感器精度的接口，这些接口可以在 Java 应用和 Java 框架中使用。

Android 传感器系统的基本层次结构如图 5-1 所示。

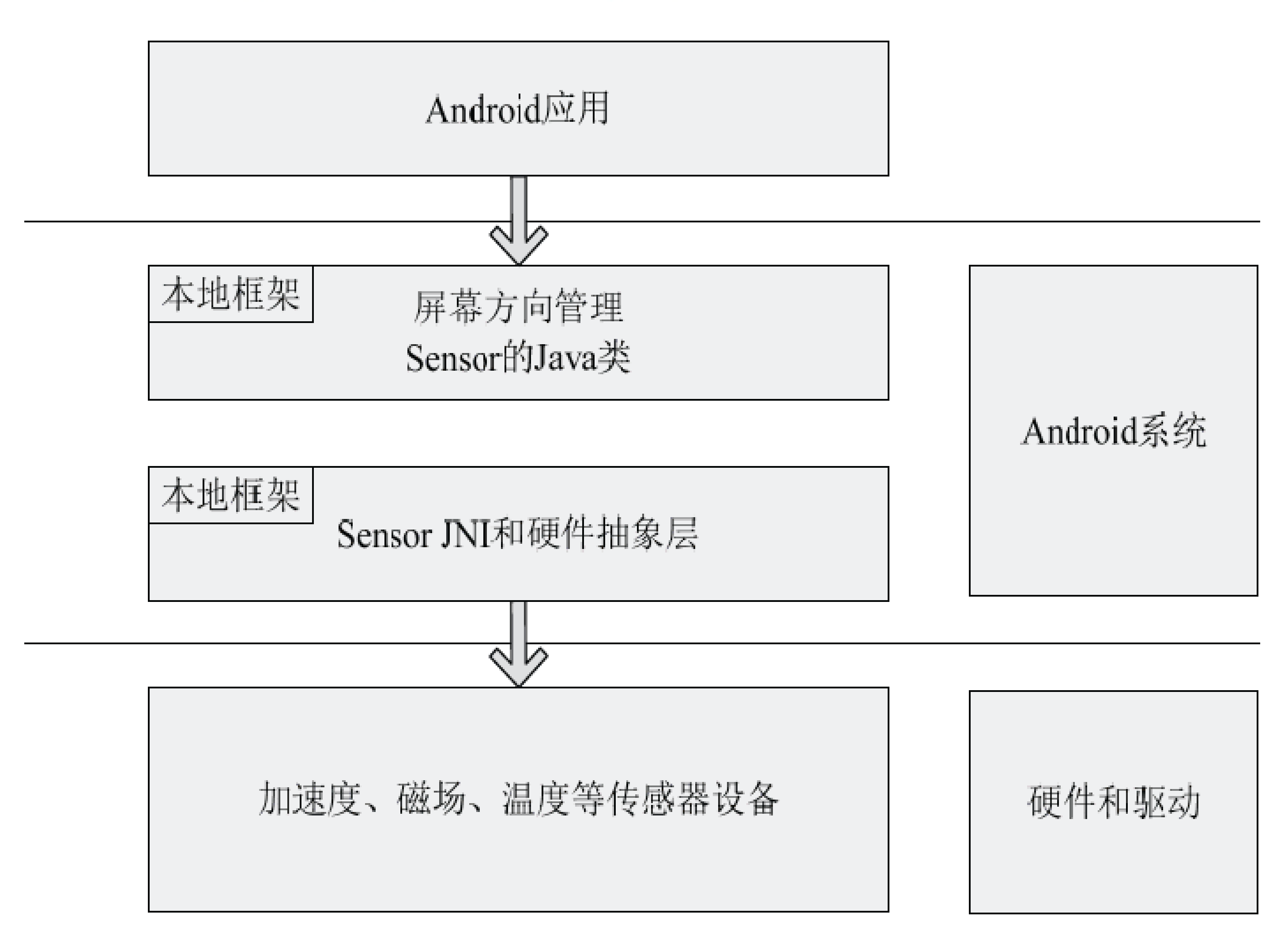


图 5-1　传感器系统的层次结构

根据图 5-1 所示的结构，Android 传感器系统从上到下分别是：Java 应用层、Java 框架对传感器的应用、传感器类、传感器硬件抽象层、传感器驱动。图 5-1 中各个层的具体说明如下。

（1）传感器系统的 Java 部分

其代码路径是：

```
frameworks/base/include/core/java/android/hardware
```

此部分对应的实现文件是 Sensor*.java。

（2）传感器系统的 JNI 部分

其代码路径是：

```
frameworks/base/core/jni/android_hardware_SensorManager.cpp
```

在此部分中提供了对类 android.hardware.Sensor.Manage 的本地支持。

（3）传感器系统 HAL 层

头文件路径是：

```
hardware/libhardware/include/hardware/sensors.h
```

在 Android 系统中，传感器系统的硬件抽象层需要特意编码实现。

（4）驱动层

驱动层的代码路径是：

```
kernel/driver/hwmon/$(PROJECT)/sensor
```

在库 sensor.so 中提供了如下 8 个 API 函数。

❑ 控制方面：在结构体ensors_control_device_t中定义，包括如下函数。
    ✧ int (*open_data_source)(struct sensors_control_device_t *dev)
    ✧ int (*activate)(struct sensors_control_device_t *dev, int handle, int enabled)
    ✧ int (*set_delay)(struct sensors_control_device_t *dev, int32_t ms)
    ✧ int (*wake)(struct sensors_control_device_t *dev)
❑ 数据方面：在结构体sensors_data_device_t中定义，包括如下函数。
    ✧ int (*data_open)(struct sensors_data_device_t *dev, int fd)
    ✧ int (*data_close)(struct sensors_data_device_t *dev)
    ✧ int (*poll)(struct sensors_data_device_t *dev, sensors_data_t* data)
❑ 模块方面：在结构体sensors_module_t中定义，包括如下函数。
    ✧ int (*get_sensors_list)(struct sensors_module_t* module, struct sensor_t const** list)

在 Android 系统的 Java 层中，Sensor 的状态是由 SensorService 来负责控制的，其 Java 代码和 JNI 代码分别位于如下文件中。

```
frameworks/base/services/java/com/android/server/SensorService.java
frameworks/base/services/jni/com_android_server_SensorService.cpp
```

SensorManager 负责在 Java 层 Sensor 的数据控制，它的 Java 代码和 JNI 代码分别位于如下文件中。

```
frameworks/base/core/java/android/hardware/SensorManager.java
frameworks/base/core/jni/android_hardware_SensorManager.cpp
```

在 Android 的 Framework 中，通过文件 sensorService.java 和 sensorManager.java 实现与 Sensor 传感器通信。文件 sensorService.java 的通信功能是通过 JNI 调用 sensorService.cpp 中的方法实现的。

文件 sensorManager.java 的具体通信功能是通过 JNI 调用 sensorManager.cpp 中的方法实现的。文件 sensorService.cpp 和 sensorManager.cpp 通过文件 hardware.c 与 sensor.so 通信。其中文件 sensorService.cpp 实现对 sensor 的状态控制，文件 sensorManger.cpp 实现对 sensor 的数据控制。库 sensor.so 通过 ioctl 控制 sensor driver 的状态，通过打开 Sensor Driver（传感器驱动）对应的设备文件读取 G-sensor 采集的数据。

在本节的内容中，将简要讲解 Android 传感器系统中各个层次的架构知识。

5.1.1　传感器系统的层详解

在 Android 系统中，传感器系统的 Java 部分的实现文件是：

```
\sdk\apps\SdkController\src\com\android\tools\sdkcontroller\activities\SensorActivity.java
```

通过阅读文件 SensorActivity.java 的源码可知，在应用程序中使用传感器需要用到 hardware 包中的 SensorManager、SensorListener 等相关的类，具体实现代码如下所示。

```
public class SensorActivity extends BaseBindingActivity
        implements android.os.Handler.Callback {

    @SuppressWarnings("hiding")
    public static String TAG = SensorActivity.class.getSimpleName();
    private static boolean DEBUG = true;

    private static final int MSG_UPDATE_ACTUAL_HZ = 0x31415;

    private TableLayout mTableLayout;
    private TextView mTextError;
    private TextView mTextStatus;
    private TextView mTextTargetHz;
    private TextView mTextActualHz;
    private SensorChannel mSensorHandler;

    private final Map<MonitoredSensor, DisplayInfo> mDisplayedSensors =
        new HashMap<SensorChannel.MonitoredSensor, SensorActivity.DisplayInfo>();
    private final android.os.Handler mUiHandler = new android.os.Handler(this);
    private int mTargetSampleRate;
    private long mLastActualUpdateMs;

    /** 第一次创建 activity 时调用 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sensors);
        mTableLayout = (TableLayout) findViewById(R.id.tableLayout);
        mTextError   = (TextView) findViewById(R.id.textError);
        mTextStatus = (TextView) findViewById(R.id.textStatus);
        mTextTargetHz = (TextView) findViewById(R.id.textSampleRate);
        mTextActualHz = (TextView) findViewById(R.id.textActualRate);
        updateStatus("Waiting for connection");

        mTextTargetHz.setOnKeyListener(new OnKeyListener() {
            @Override
            public boolean onKey(View v, int keyCode, KeyEvent event) {
                updateSampleRate();
                return false;
            }
        });
        mTextTargetHz.setOnFocusChangeListener(new OnFocusChangeListener() {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                updateSampleRate();
            }
        });
```

```
    }

    @Override
    protected void onResume() {
        if (DEBUG) Log.d(TAG, "onResume");
        // BaseBindingActivity 绑定后套服务
        super.onResume();
        updateError();
    }

    @Override
    protected void onPause() {
        if (DEBUG) Log.d(TAG, "onPause");
        super.onPause();
    }

    @Override
    protected void onDestroy() {
        if (DEBUG) Log.d(TAG, "onDestroy");
        super.onDestroy();
        removeSensorUi();
    }
//创建传感器 UI
    private void createSensorUi() {
        final LayoutInflater inflater = getLayoutInflater();

        if (!mDisplayedSensors.isEmpty()) {
            removeSensorUi();
        }

        mSensorHandler = (SensorChannel) getServiceBinder().getChannel(Channel.SENSOR_CHANNEL);
        if (mSensorHandler != null) {
            mSensorHandler.addUiHandler(mUiHandler);
            mUiHandler.sendEmptyMessage(MSG_UPDATE_ACTUAL_HZ);

            assert mDisplayedSensors.isEmpty();
            List<MonitoredSensor> sensors = mSensorHandler.getSensors();
            for (MonitoredSensor sensor : sensors) {
                final TableRow row = (TableRow) inflater.inflate(R.layout.sensor_row,
                                                                mTableLayout,
                                                                false);
                mTableLayout.addView(row);
                mDisplayedSensors.put(sensor, new DisplayInfo(sensor, row));
            }
        }
    }
//删除传感器 UI
    private void removeSensorUi() {
        if (mSensorHandler != null) {
            mSensorHandler.removeUiHandler(mUiHandler);
            mSensorHandler = null;
```

```
        }
        mTableLayout.removeAllViews();
        for (DisplayInfo info : mDisplayedSensors.values()) {
            info.release();
        }
        mDisplayedSensors.clear();
    }

    private class DisplayInfo implements CompoundButton.OnCheckedChangeListener {
        private MonitoredSensor mSensor;
        private CheckBox mChk;
        private TextView mVal;

        public DisplayInfo(MonitoredSensor sensor, TableRow row) {
            mSensor = sensor;

            // Initialize displayed checkbox for this sensor, and register
            // checked state listener for it
            mChk = (CheckBox) row.findViewById(R.id.row_checkbox);
            mChk.setText(sensor.getUiName());
            mChk.setEnabled(sensor.isEnabledByEmulator());
            mChk.setChecked(sensor.isEnabledByUser());
            mChk.setOnCheckedChangeListener(this);

            //初始化显示该传感器的文本框
            mVal = (TextView) row.findViewById(R.id.row_textview);
            mVal.setText(sensor.getValue());
        }

        /**
         *为相关的复选框选中状态进行变化的处理。当复选框被选中时会注册传感器变化
         *如果不加以控制会取消传感器的变化
         */
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (mSensor != null) {
                mSensor.onCheckedChanged(isChecked);
            }
        }

        public void release() {
            mChk = null;
            mVal = null;
            mSensor = null;

        }

        public void updateState() {
            if (mChk != null && mSensor != null) {
                mChk.setEnabled(mSensor.isEnabledByEmulator());
                mChk.setChecked(mSensor.isEnabledByUser());
```

```
            }
        }

        public void updateValue() {
            if (mVal != null && mSensor != null) {
                mVal.setText(mSensor.getValue());
            }
        }
    }

    /**实现回调处理程序*/
    @Override
    public boolean handleMessage(Message msg) {
        DisplayInfo info = null;
        switch (msg.what) {
        case SensorChannel.SENSOR_STATE_CHANGED:
            info = mDisplayedSensors.get(msg.obj);
            if (info != null) {
                info.updateState();
            }
            break;
        case SensorChannel.SENSOR_DISPLAY_MODIFIED:
            info = mDisplayedSensors.get(msg.obj);
            if (info != null) {
                info.updateValue();
            }
            if (mSensorHandler != null) {
                updateStatus(Integer.toString(mSensorHandler.getMsgSentCount()) + " events sent");

                //如果值已经修改则更新 "actual rate"
                long ms = mSensorHandler.getActualUpdateMs();
                if (ms != mLastActualUpdateMs) {
                    mLastActualUpdateMs = ms;
                    String hz = mLastActualUpdateMs <= 0 ? "--" :
                                    Integer.toString((int) Math.ceil(1000. / ms));
                    mTextActualHz.setText(hz);
                }
            }
            break;
        case MSG_UPDATE_ACTUAL_HZ:
            if (mSensorHandler != null) {
                //如果值已经修改则更新 "actual rate"
                long ms = mSensorHandler.getActualUpdateMs();
                if (ms != mLastActualUpdateMs) {
                    mLastActualUpdateMs = ms;
                    String hz = mLastActualUpdateMs <= 0 ? "--" :
                                    Integer.toString((int) Math.ceil(1000. / ms));
                    mTextActualHz.setText(hz);
                }
                mUiHandler.sendEmptyMessageDelayed(MSG_UPDATE_ACTUAL_HZ, 1000 /*1s*/);
            }
```

```
            }
            return true; // we consumed this message
        }
        private void updateSampleRate() {
            String str = mTextTargetHz.getText().toString();
            try {
                int hz = Integer.parseInt(str.trim());

                //上限值 50，模拟器的最大值是 50 赫兹
                if (hz <= 0 || hz > 50) {
                    hz = 50;
                }

                if (hz != mTargetSampleRate) {
                    mTargetSampleRate = hz;
                    if (mSensorHandler != null) {
                        mSensorHandler.setUpdateTargetMs(hz <= 0 ? 0 : (int)(1000.0f / hz));
                    }
                }
            } catch (Exception ignore) { }
        }
}
```

通过上述代码可知，整个 Java 层利用观察者模式对传感器的数据进行了监听处理。

5.1.2 Frameworks 层详解

在 Android 系统中，Frameworks 层是 Android 系统提供的应用程序开发接口和应用程序框架，与应用程序的调用是通过类实例化或类继承进行的。对应用程序来说，最重要的就是把 SensorListener 注册到 SensorManager 上，从而才能以观察者身份接收到数据的变化，因此，我们把目光落在 SensorManager 的构造函数、RegisterListener 函数和通知机制相关的代码上。在 Android 传感器系统中，Frameworks 层的代码路径是 frameworks/base/include/core/java/android/hardware。在本节的内容中，将详细讲解传感器系统的 Frameworks 层的具体实现流程。

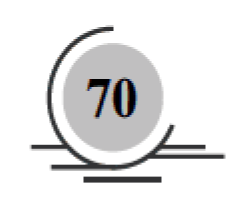
1. 监听传感器的变化

在 Android 传感器系统的 Frameworks 层中，文件 SensorListener.java 用于监听从 Java 应用层中传递过来的变化。文件 SensorListener.java 比较简单，具体代码如下所示。

```
package android.hardware;
@Deprecated
public interface SensorListener {
    public void onSensorChanged(int sensor, float[] values);
    public void onAccuracyChanged(int sensor, int accuracy);
}
```

2. 注册监听

当文件 SensorListener.java 监听到变化之后，会通过文件 SensorManager.java 向服务注册监听变化，并调度 Sensor 的具体任务。例如在开发 Android 传感器应用程序时，在上层的通用开发流程如下。



（1）通过“getSystemService(SENSOR_SERVICE);”语句得到传感器服务。这样得到一个用来管理分配调度处理 Sensor 工作的 SensorManager。SensorManager 并不服务运行于后台，真正属于 Sensor 的系统服务是 SensorService，在终端下的#service list 中可以看到 sensorservice: [android.gui. SensorServer]。

（2）通过“getDefaultSensor(Sensor.TYPE_GRAVITY);”得到传感器类型，当然还有各种千奇百怪的传感器，具体可以查阅 Android 官网的 API 或者源码中的文件 Sensor.java。

（3）注册监听器 SensorEventListener。在应用程序中打开一个监听接口，专门用于处理传感器的数据。

（4）通过回调函数 onSensorChanged 和 onAccuracyChanged 实现实时监听。例如对重力感应器的 xyz 值经算法变换得到左右上下前后方向等，就由这个回调函数实现。

综上所述，传感器顶层的处理流程如图 5-2 所示。

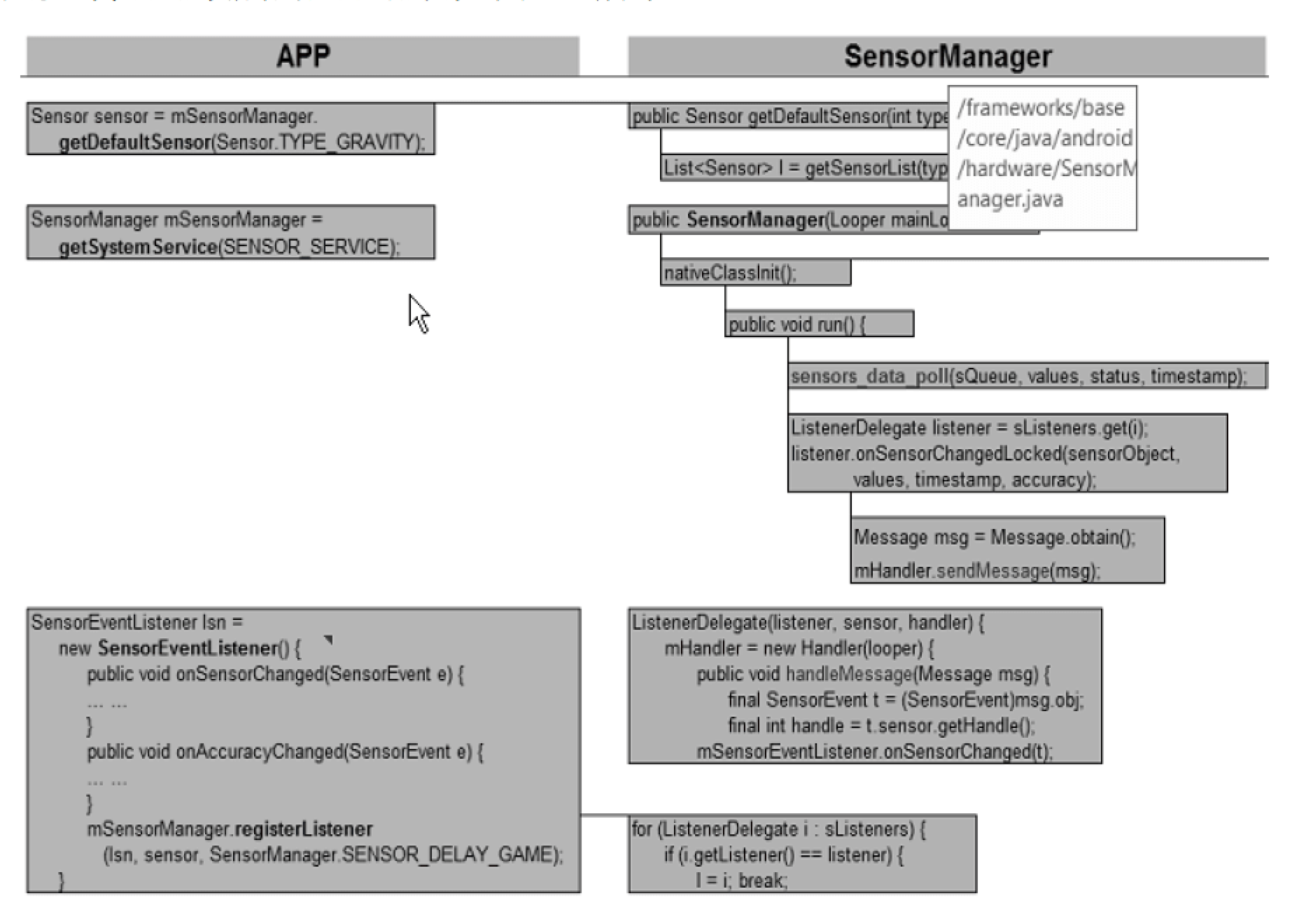


图 5-2 传感器顶层的处理流程

文件 SensorManager.java 的具体实现流程如下。

（1）定义类 SensorManager，然后设置各种传感器的初始变量值，具体代码如下所示。

```
public abstract class SensorManager {
    protected static final String TAG = "SensorManager";
    private static final float[ ] mTempMatrix = new float[16];

    // Cached lists of sensors by type.  Guarded by mSensorListByType
    private final SparseArray<List<Sensor>> mSensorListByType =
            new SparseArray<List<Sensor>>();

    // Legacy sensor manager implementation.  Guarded by mSensorListByType during initialization
    private LegacySensorManager mLegacySensorManager;
    @Deprecated
    public static final int SENSOR_ORIENTATION = 1 << 0;
    @Deprecated
    public static final int SENSOR_ACCELEROMETER = 1 << 1;
```

```
@Deprecated
public static final int SENSOR_TEMPERATURE = 1 << 2;
@Deprecated
public static final int SENSOR_MAGNETIC_FIELD = 1 << 3;
@Deprecated
public static final int SENSOR_LIGHT = 1 << 4;
@Deprecated
public static final int SENSOR_PROXIMITY = 1 << 5;
@Deprecated
public static final int SENSOR_TRICORDER = 1 << 6;
@Deprecated
public static final int SENSOR_ORIENTATION_RAW = 1 << 7;
@Deprecated
public static final int SENSOR_ALL = 0x7F;
@Deprecated
public static final int SENSOR_MIN = SENSOR_ORIENTATION;
@Deprecated
public static final int SENSOR_MAX = ((SENSOR_ALL + 1)>>1);

@Deprecated
public static final int DATA_X = 0;
@Deprecated
public static final int DATA_Y = 1;
@Deprecated
public static final int DATA_Z = 2;
@Deprecated
public static final int RAW_DATA_INDEX = 3;
@Deprecated
public static final int RAW_DATA_X = 3;
@Deprecated
public static final int RAW_DATA_Y = 4;
@Deprecated
public static final int RAW_DATA_Z = 5;

public static final float STANDARD_GRAVITY = 9.80665f;

public static final float GRAVITY_SUN              = 275.0f;
public static final float GRAVITY_MERCURY          = 3.70f;

public static final float GRAVITY_VENUS            = 8.87f;
public static final float GRAVITY_EARTH            = 9.80665f;
public static final float GRAVITY_MOON             = 1.6f;
public static final float GRAVITY_MARS             = 3.71f;
public static final float GRAVITY_JUPITER          = 23.12f;
public static final float GRAVITY_SATURN           = 8.96f;
public static final float GRAVITY_URANUS           = 8.69f;
public static final float GRAVITY_NEPTUNE          = 11.0f;
public static final float GRAVITY_PLUTO            = 0.6f;
public static final float GRAVITY_DEATH_STAR_I     = 0.000000353036145f;
public static final float GRAVITY_THE_ISLAND       = 4.815162342f;
```

```
/**对地球表面的最大磁场*/
public static final float MAGNETIC_FIELD_EARTH_MAX = 60.0f;
/**对地球表面的最小磁场*/
public static final float MAGNETIC_FIELD_EARTH_MIN = 30.0f;
/** 标准大气压 */
public static final float PRESSURE_STANDARD_ATMOSPHERE = 1013.25f;
public static final float LIGHT_SUNLIGHT_MAX = 120000.0f;
public static final float LIGHT_SUNLIGHT        = 110000.0f;
public static final float LIGHT_SHADE           = 20000.0f;
public static final float LIGHT_OVERCAST        = 10000.0f;
public static final float LIGHT_SUNRISE         = 400.0f;
public static final float LIGHT_CLOUDY          = 100.0f;
public static final float LIGHT_FULLMOON        = 0.25f;
public static final float LIGHT_NO_MOON         = 0.001f;


/**尽可能快地获得传感器数据*/
public static final int SENSOR_DELAY_FASTEST = 0;
/**适合游戏速度*/
public static final int SENSOR_DELAY_GAME = 1;
/**适合于用户接口速率*/
public static final int SENSOR_DELAY_UI = 2;
/**（默认值）适合屏幕方向的变化*/
public static final int SENSOR_DELAY_NORMAL = 3;
/**
 *返回的值，该传感器是不可信的，需要进行校准或环境不允许读数
 */
public static final int SENSOR_STATUS_UNRELIABLE = 0;
/**
 *该传感器是报告的低精度的数据，与环境的校准是必要的
 */
public static final int SENSOR_STATUS_ACCURACY_LOW = 1;
/**
 *这种传感器是精确的平均频率的报告数据，与环境的校准可以提高精确度
 */
public static final int SENSOR_STATUS_ACCURACY_MEDIUM = 2;

/**该传感报告准确性最大的数据*/
public static final int SENSOR_STATUS_ACCURACY_HIGH = 3;
public static final int AXIS_X = 1;
public static final int AXIS_Y = 2;
public static final int AXIS_Z = 3;
public static final int AXIS_MINUS_X = AXIS_X | 0x80;
public static final int AXIS_MINUS_Y = AXIS_Y | 0x80;
public static final int AXIS_MINUS_Z = AXIS_Z | 0x80;
```

（2）定义各种设备类型和设备数据的方法，这些方法非常重要，在编写的应用程序中，可以通过 AIDL 接口远程调用（RPC）的方式得到 SensorManager。这样通过在类 SensorManager 中的方法，可以得到底层的各种传感器数据。上述方法的具体实现代码如下所示。

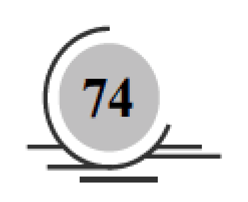
```
    public int getSensors() {
        return getLegacySensorManager().getSensors();
    }
    public List<Sensor> getSensorList(int type) {
        //第一次缓存返回列表
        List<Sensor> list;
        final List<Sensor> fullList = getFullSensorList();
        synchronized (mSensorListByType) {
            list = mSensorListByType.get(type);
            if (list == null) {
                if (type == Sensor.TYPE_ALL) {
                    list = fullList;
                } else {
                    list = new ArrayList<Sensor>();
                    for (Sensor i : fullList) {
                        if (i.getType() == type)
                            list.add(i);
                    }
                }
                list = Collections.unmodifiableList(list);
                mSensorListByType.append(type, list);
            }
        }
        return list;
    }
…
```

上述方法的功能非常重要，其实就是我们在开发传感器应用程序时用到的 API 接口。有关上述方法的具体说明，读者可以查阅官网 SDK API 中对于类 android.hardware.SensorManager 的具体说明，如图 5-3 所示。

- Introduction
- App Components
- App Resources
- App Manifest
- User Interface
- Animation and Graphics
- Computation
- Media and Camera
- Location and Sensors
- Connectivity
- Text and Input
- Data Storage
- Administration

Sensor	Type	Description	Common Uses
TYPE_ACCELEROMETER	Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
TYPE_AMBIENT_TEMPERATURE	Hardware	Measures the ambient room temperature in degrees Celsius (°C). See note below.	Monitoring air temperatures.
TYPE_GRAVITY	Software or Hardware	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
TYPE_GYROSCOPE	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
TYPE_LIGHT	Hardware	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
TYPE_LINEAR_ACCELERATION	Software or Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of	Monitoring acceleration along a single

图 5-3　Android SDK API 中对 android.hardware.SensorManager 的具体说明



5.1.3　JNI 层详解

在 Android 系统中，传感器系统的 JNI 部分的代码路径是：

```
frameworks/base/core/jni/android_hardware_SensorManager.cpp
```

在此文件中提供了对类 android.hardware.Sensor.Manager 的本地支持。上层和 JNI 层的调用关系如图 5-4 所示。

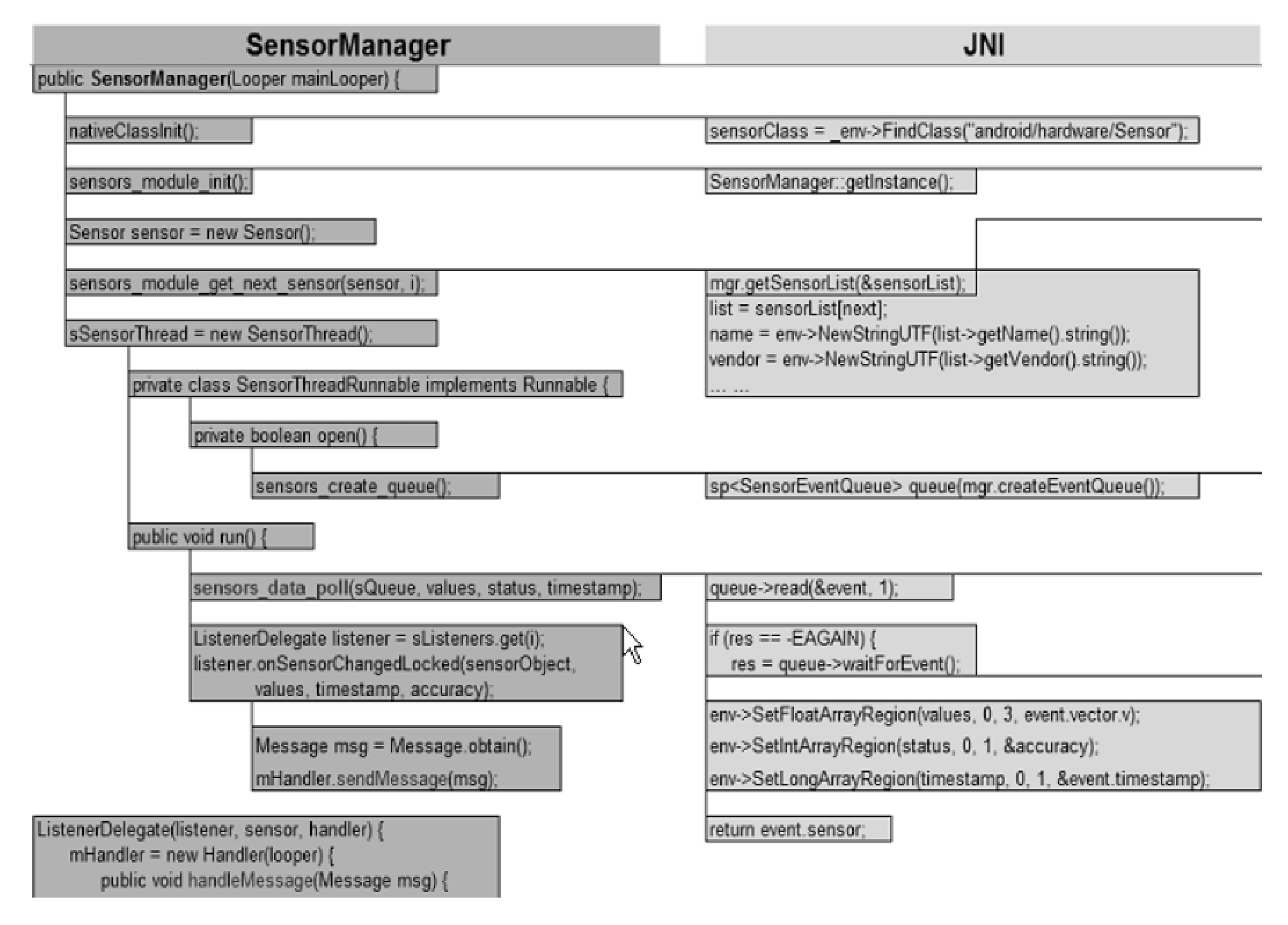


图 5-4　上层和 JNI 层的调用关系

在图 5-4 所示的调用关系中涉及了如下的 API 接口方法。

- nativeClassInit()：在JNI层得到android.hardware.Sensor的JNI环境指针。
- sensors_module_init()：通过JNI调用本地框架，得到SensorService，SensorService初始化控制流各功能。
- new Sensor()：建立一个Sensor对象，具体可查阅官网API android.hardware.Sensor。
- sensors_module_get_next_sensor()：上层得到设备支持的所有Sensor，并放入SensorList链表。
- new SensorThread()：创建Sensor线程，当应用程序registerListener()注册监听器的时候开启线程run()，注意当没有数据变化时线程会阻塞。

1. 实现本地函数

文件 android_hardware_SensorManager.cpp 的功能是实现文件 SensorManager.java 中的 native（本地）函数，主要是通过调用文件 SensorManager.cpp 和文件 SensorEventQueue.cpp 中的相关类来完成相关的工作。文件 android_hardware_SensorManager.cpp 的具体实现代码如下所示。

```
static struct {
    jclass clazz;
    jmethodID dispatchSensorEvent;
} gBaseEventQueueClassInfo;
```

```
namespace android {

struct SensorOffsets
{
    jfieldID    name;
    jfieldID    vendor;
    jfieldID    version;
    jfieldID    handle;
    jfieldID    type;
    jfieldID    range;
    jfieldID    resolution;
    jfieldID    power;
    jfieldID    minDelay;
} gSensorOffsets;


/*
 *下面的方法是非线程安全的和不打算用的
 */

static void
nativeClassInit (JNIEnv *_env, jclass _this)
{
    jclass sensorClass = _env->FindClass("android/hardware/Sensor");
    SensorOffsets& sensorOffsets = gSensorOffsets;
    sensorOffsets.name          = _env->GetFieldID(sensorClass, "mName", "Ljava/lang/String;");
    sensorOffsets.vendor        = _env->GetFieldID(sensorClass, "mVendor", "Ljava/lang/String;");
    sensorOffsets.version       = _env->GetFieldID(sensorClass, "mVersion", "I");
    sensorOffsets.handle        = _env->GetFieldID(sensorClass, "mHandle", "I");
    sensorOffsets.type          = _env->GetFieldID(sensorClass, "mType", "I");
    sensorOffsets.range         = _env->GetFieldID(sensorClass, "mMaxRange", "F");
    sensorOffsets.resolution    = _env->GetFieldID(sensorClass, "mResolution","F");
    sensorOffsets.power         = _env->GetFieldID(sensorClass, "mPower", "F");
    sensorOffsets.minDelay      = _env->GetFieldID(sensorClass, "mMinDelay", "I");
}

static jint
nativeGetNextSensor(JNIEnv *env, jclass clazz, jobject sensor, jint next)
{
    SensorManager& mgr(SensorManager::getInstance());

    Sensor const* const* sensorList;
    size_t count = mgr.getSensorList(&sensorList);
    if (size_t(next) >= count)
        return -1;

    Sensor const* const list = sensorList[next];
    const SensorOffsets& sensorOffsets(gSensorOffsets);
    jstring name = env->NewStringUTF(list->getName().string());
    jstring vendor = env->NewStringUTF(list->getVendor().string());
    env->SetObjectField(sensor, sensorOffsets.name, name);
```

```
    env->SetObjectField(sensor, sensorOffsets.vendor, vendor);
    env->SetIntField(sensor, sensorOffsets.version, list->getVersion());
    env->SetIntField(sensor, sensorOffsets.handle, list->getHandle());
    env->SetIntField(sensor, sensorOffsets.type, list->getType());
    env->SetFloatField(sensor, sensorOffsets.range, list->getMaxValue());
    env->SetFloatField(sensor, sensorOffsets.resolution, list->getResolution());
    env->SetFloatField(sensor, sensorOffsets.power, list->getPowerUsage());
    env->SetIntField(sensor, sensorOffsets.minDelay, list->getMinDelay());

    next++;
    return size_t(next) < count ? next : 0;
}
```

2. 处理客户端数据

文件 frameworks\native\libs\gui\SensorManager.cpp 功能是提供了对传感器数据部分的操作，实现了 sensor_data_XXX()格式的函数。另外在 Native 层的客户端，文件 SensorManager.cpp 还负责与服务端 SensorService.cpp 之间的通信工作。文件 SensorManager.cpp 的具体实现代码如下所示。

```
// ----------------------------------------------------------------------------
namespace android {
// ----------------------------------------------------------------------------

ANDROID_SINGLETON_STATIC_INSTANCE(SensorManager)

SensorManager::SensorManager()
    : mSensorList(0)
{
    // we're not locked here, but it's not needed during construction
    assertStateLocked();
}

SensorManager::~SensorManager()
{
    free(mSensorList);
}

void SensorManager::sensorManagerDied()
{
    Mutex::Autolock _l(mLock);
    mSensorServer.clear();
    free(mSensorList);
    mSensorList = NULL;
    mSensors.clear();
}
```

3. 处理服务端数据

文件 frameworks\native\services\sensorservice\SensorService.cpp 能够实现 Sensor 真正的后台服务，是服务端的数据处理中心。在 Android 的传感器系统中，SensorService 作为一个轻量级的 System Service，在 SystemServer 内运行，在 system_init<system_init.cpp>中调用了 SensorService::instantiate()。具体来说，

SensorService 的主要功能如下。

（1）通过 SensorService::instantiate 创建实例对象，并增加到 ServiceManager 中，然后创建并启动线程，并执行 threadLoop。

（2）threadLoop 从 sensor 驱动获取原始数据，然后通过 SensorEventConnection 把事件发送给客户端。

（3）BnSensorServer 的成员函数负责让客户端获取 sensor 列表和创建 SensorEventConnection。

文件 SensorService.cpp 的具体实现代码如下所示。

```
namespace android {

const char* SensorService::WAKE_LOCK_NAME = "SensorService";

SensorService::SensorService()
    : mInitCheck(NO_INIT)
{
}

void SensorService::onFirstRef()
{
    ALOGD("nuSensorService starting...");

    SensorDevice& dev(SensorDevice::getInstance());

    if (dev.initCheck() == NO_ERROR) {
        sensor_t const* list;
        ssize_t count = dev.getSensorList(&list);
        if (count > 0) {
            ssize_t orientationIndex = -1;
            bool hasGyro = false;
            uint32_t virtualSensorsNeeds =
                    (1<<SENSOR_TYPE_GRAVITY) |
                    (1<<SENSOR_TYPE_LINEAR_ACCELERATION) |
                    (1<<SENSOR_TYPE_ROTATION_VECTOR);

            mLastEventSeen.setCapacity(count);
            for (ssize_t i=0 ; i<count ; i++) {
                registerSensor( new HardwareSensor(list[i]) );
                switch (list[i].type) {
                    case SENSOR_TYPE_ORIENTATION:
                        orientationIndex = i;
                        break;
                    case SENSOR_TYPE_GYROSCOPE:
                        hasGyro = true;
                        break;
                    case SENSOR_TYPE_GRAVITY:
                    case SENSOR_TYPE_LINEAR_ACCELERATION:
                    case SENSOR_TYPE_ROTATION_VECTOR:
                        virtualSensorsNeeds &= ~(1<<list[i].type);
                        break;
                }
            }
```

```
//它是安全的，在这里实例化 SensorFusion 对象
//如果要被实例化后，H/W 传感器已注册
const SensorFusion& fusion(SensorFusion::getInstance());

if (hasGyro) {
    //总是实例化 Android 的虚拟传感器。因为它们是实例化落后于 HAL 的传感器，它们不会干扰
    应用程序，除非它们看起来特别像它们的名字

    registerVirtualSensor( new RotationVectorSensor() );
    registerVirtualSensor( new GravitySensor(list, count) );
    registerVirtualSensor( new LinearAccelerationSensor(list, count) );

    // 这是选项
    registerVirtualSensor( new OrientationSensor() );
    registerVirtualSensor( new CorrectedGyroSensor(list, count) );
}

// build the sensor list returned to users
mUserSensorList = mSensorList;

if (hasGyro) {
    // virtual debugging sensors are not added to mUserSensorList
    registerVirtualSensor( new GyroDriftSensor() );
}

if (hasGyro &&
        (virtualSensorsNeeds & (1<<SENSOR_TYPE_ROTATION_VECTOR))) {
    // if we have the fancy sensor fusion, and it's not provided by the
    // HAL, use our own (fused) orientation sensor by removing the
    // HAL supplied one form the user list.
    if (orientationIndex >= 0) {
        mUserSensorList.removeItemsAt(orientationIndex);
    }
}

//调试传感器列表
for (size_t i=0 ; i<mSensorList.size() ; i++) {
    switch (mSensorList[i].getType()) {
        case SENSOR_TYPE_GRAVITY:
        case SENSOR_TYPE_LINEAR_ACCELERATION:
        case SENSOR_TYPE_ROTATION_VECTOR:
            if (strstr(mSensorList[i].getVendor().string(), "Google")) {
                mUserSensorListDebug.add(mSensorList[i]);
            }
            break;
        default:
            mUserSensorListDebug.add(mSensorList[i]);
            break;
    }
}
```

```
                run("SensorService", PRIORITY_URGENT_DISPLAY);
                mInitCheck = NO_ERROR;
            }
        }
}

void SensorService::registerSensor(SensorInterface* s)
{
    sensors_event_t event;
    memset(&event, 0, sizeof(event));

    const Sensor sensor(s->getSensor());
    //添加到传感器列表（返回给客户端）
    mSensorList.add(sensor);
    //加入到我们的手柄 - > SensorInterface 映射
    mSensorMap.add(sensor.getHandle(), s);
    //创建 mLastEventSeen 数组中的一个条目
    mLastEventSeen.add(sensor.getHandle(), event);
}

void SensorService::registerVirtualSensor(SensorInterface* s)
{
    registerSensor(s);
    mVirtualSensorList.add( s );
}

SensorService::~SensorService()
{
    for (size_t i=0 ; i<mSensorMap.size() ; i++)
        delete mSensorMap.valueAt(i);
}
…
```

通过上述实现代码，可以了解 SensorService 服务的创建、启动过程，整个过程的 C/S 通信架构如图 5-5 所示。

在此需要注意，并没有在系统中使用 BpSensorServer，即使从 ISensorServer.cpp 中把它删除也不会对 Sensor 的工作有任何影响。这是因为它的工作已经被文件 SensorManager.cpp 所取代，ServiceManager 会直接获取上面文件 System_init 中添加的 SensorService 对象。

4. 封装 HAL 层的代码

在 Android 系统中，通过文件 frameworks\native\services\sensorservice\SensorDevice.cpp 封装了 HAL 层的代码，此文件的主要功能如下：

- 获取sensor列表（getSensorList）。
- 获取sensor事件（poll）。
- Enable或Disable sensor (activate)。
- 设置delay时间。

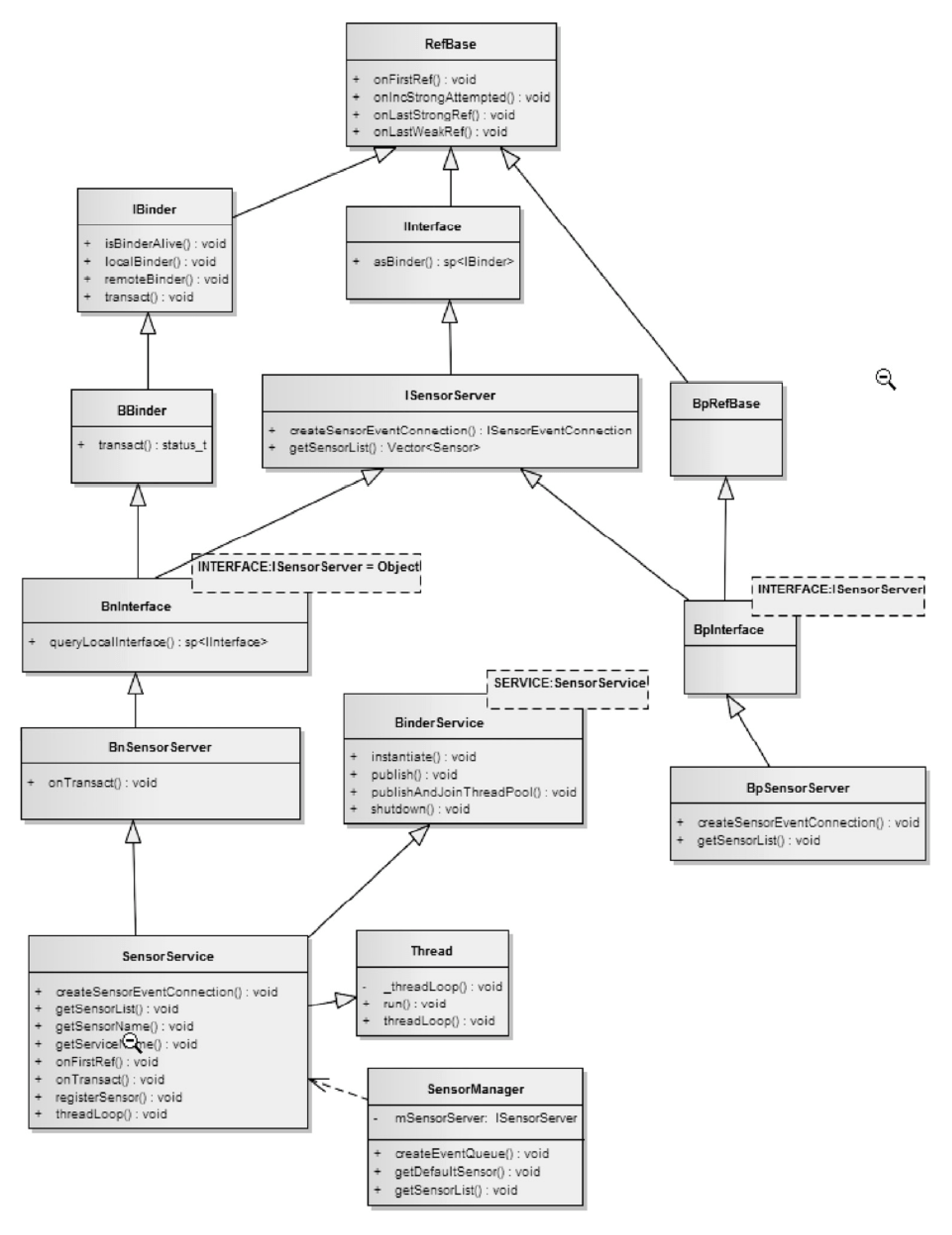


图 5-5　C/S 通信架构图

文件 SensorDevice.cpp 的主要实现代码如下所示。

```
status_t SensorDevice::activate(void* ident, int handle, int enabled)
{
    if (!mSensorDevice) return NO_INIT;
    status_t err(NO_ERROR);
    bool actuateHardware = false;

    Info& info( mActivationCount.editValueFor(handle) );


    ALOGD_IF(DEBUG_CONNECTIONS,
```

```
            "SensorDevice::activate: ident=%p, handle=0x%08x, enabled=%d, count=%d",
            ident, handle, enabled, info.rates.size());

    if (enabled) {
        Mutex::Autolock _l(mLock);
        ALOGD_IF(DEBUG_CONNECTIONS, "... index=%ld",
                info.rates.indexOfKey(ident));

        if (info.rates.indexOfKey(ident) < 0) {
            info.rates.add(ident, DEFAULT_EVENTS_PERIOD);
            if (info.rates.size() == 1) {
                actuateHardware = true;
            }
        } else {
            //传感器已经激活此 IDENT
        }
    } else {
        Mutex::Autolock _l(mLock);
        ALOGD_IF(DEBUG_CONNECTIONS, "... index=%ld",
                info.rates.indexOfKey(ident));

        ssize_t idx = info.rates.removeItem(ident);
        if (idx >= 0) {
            if (info.rates.size() == 0) {
                actuateHardware = true;
            }
        } else {
            //没有启用这个传感器的 ident
        }
    }

    if (actuateHardware) {
        ALOGD_IF(DEBUG_CONNECTIONS, "\t>>> actuating h/w");

        err = mSensorDevice->activate(mSensorDevice, handle, enabled);
        ALOGE_IF(err, "Error %s sensor %d (%s)",
                enabled ? "activating" : "disabling",
                handle, strerror(-err));
    }

    { //范围为锁
        Mutex::Autolock _l(mLock);
        nsecs_t ns = info.selectDelay();
        mSensorDevice->setDelay(mSensorDevice, handle, ns);
    }

    return err;
}

status_t SensorDevice::setDelay(void* ident, int handle, int64_t ns)
{
```

```
    if (!mSensorDevice) return NO_INIT;
    Mutex::Autolock _l(mLock);
    Info& info( mActivationCount.editValueFor(handle) );
    status_t err = info.setDelayForIdent(ident, ns);
    if (err < 0) return err;
    ns = info.selectDelay();
    return mSensorDevice->setDelay(mSensorDevice, handle, ns);
}

int SensorDevice::getHalDeviceVersion() const {
    if (!mSensorDevice) return -1;

    return mSensorDevice->common.version;
}

// ---------------------------------------------------------------------------

status_t SensorDevice::Info::setDelayForIdent(void* ident, int64_t ns)
{
    ssize_t index = rates.indexOfKey(ident);
    if (index < 0) {
        ALOGE("Info::setDelayForIdent(ident=%p, ns=%lld) failed (%s)",
                ident, ns, strerror(-index));
        return BAD_INDEX;
    }
    rates.editValueAt(index) = ns;
    return NO_ERROR;
}

nsecs_t SensorDevice::Info::selectDelay()
{
    nsecs_t ns = rates.valueAt(0);
    for (size_t i=1 ; i<rates.size() ; i++) {
        nsecs_t cur = rates.valueAt(i);
        if (cur < ns) {
            ns = cur;
        }
    }
    delay = ns;
    return ns;
}

// ---------------------------------------------------------------------------
}; // namespace android
```

这样 SensorService 会把任务交给 SensorDevice，而 SensorDevice 会调用标准的抽象层接口。由此可见，Sensor 架构的抽象层接口是最标准的一种，它很好地实现了抽象层与本地框架的分离。

5. 处理消息队列

在 Android 传感器系统中，文件 frameworks\native\libs\gui\SensorEventQueue.cpp 的功能是处理消息。

文件 SensorEventQueue.cpp 能够在创建其实例时传入 SensorEventConnection 实例，SensorEventConnection 继承于 ISensorEventConnection。SensorEventConnection 其实是客户端调用 SensorService 的 createSensorEventConnection()方法创建的，是客户端与服务端沟通的桥梁，通过这个桥梁可以完成如下功能：

- 获取管道的句柄。
- 往管道读写数据。
- 通知服务端对Sensor使能。

文件 frameworks\native\libs\gui\SensorEventQueue.cpp 的具体实现代码如下所示。

```
// ----------------------------------------------------------------------------
namespace android {
// ----------------------------------------------------------------------------

SensorEventQueue::SensorEventQueue(const sp<ISensorEventConnection>& connection)
    : mSensorEventConnection(connection)
{
}

SensorEventQueue::~SensorEventQueue()
{
}

void SensorEventQueue::onFirstRef()
{
    mSensorChannel = mSensorEventConnection->getSensorChannel();
}

int SensorEventQueue::getFd() const
{
    return mSensorChannel->getFd();
}


ssize_t SensorEventQueue::write(const sp<BitTube>& tube,
        ASensorEvent const* events, size_t numEvents) {
    return BitTube::sendObjects(tube, events, numEvents);
}

ssize_t SensorEventQueue::read(ASensorEvent* events, size_t numEvents)
{
    return BitTube::recvObjects(mSensorChannel, events, numEvents);
}

sp<Looper> SensorEventQueue::getLooper() const
{
    Mutex::Autolock _l(mLock);
    if (mLooper == 0) {
        mLooper = new Looper(true);
        mLooper->addFd(getFd(), getFd(), ALOOPER_EVENT_INPUT, NULL, NULL);
```

```
    }
    return mLooper;
}

status_t SensorEventQueue::waitForEvent() const
{
    const int fd = getFd();
    sp<Looper> looper(getLooper());

    int events;
    int32_t result;
    do {
        result = looper->pollOnce(-1, NULL, &events, NULL);
        if (result == ALOOPER_POLL_ERROR) {
            ALOGE("SensorEventQueue::waitForEvent error (errno=%d)", errno);
            result = -EPIPE; // unknown error, so we make up one
            break;
        }
        if (events & ALOOPER_EVENT_HANGUP) {
            // the other-side has died
            ALOGE("SensorEventQueue::waitForEvent error HANGUP");
            result = -EPIPE; // unknown error, so we make up one
            break;
        }
    } while (result != fd);

    return   (result == fd) ? status_t(NO_ERROR) : result;
}

status_t SensorEventQueue::wake() const
{
    sp<Looper> looper(getLooper());
    looper->wake();
    return NO_ERROR;
}

status_t SensorEventQueue::enableSensor(Sensor const* sensor) const {
    return mSensorEventConnection->enableDisable(sensor->getHandle(), true);
}

status_t SensorEventQueue::disableSensor(Sensor const* sensor) const {
    return mSensorEventConnection->enableDisable(sensor->getHandle(), false);
}

status_t SensorEventQueue::enableSensor(int32_t handle, int32_t us) const {
    status_t err = mSensorEventConnection->enableDisable(handle, true);
    if (err == NO_ERROR) {
        mSensorEventConnection->setEventRate(handle, us2ns(us));
    }
    return err;
}
```

```
status_t SensorEventQueue::disableSensor(int32_t handle) const {
    return mSensorEventConnection->enableDisable(handle, false);
}

status_t SensorEventQueue::setEventRate(Sensor const* sensor, nsecs_t ns) const {
    return mSensorEventConnection->setEventRate(sensor->getHandle(), ns);
}

// ----------------------------------------------------------------------
}; // namespace android
```

由此可见，SensorManager 负责控制流，通过 C/S 的 Binder 机制与 SensorService 实现通信。具体过程如图 5-6 所示。

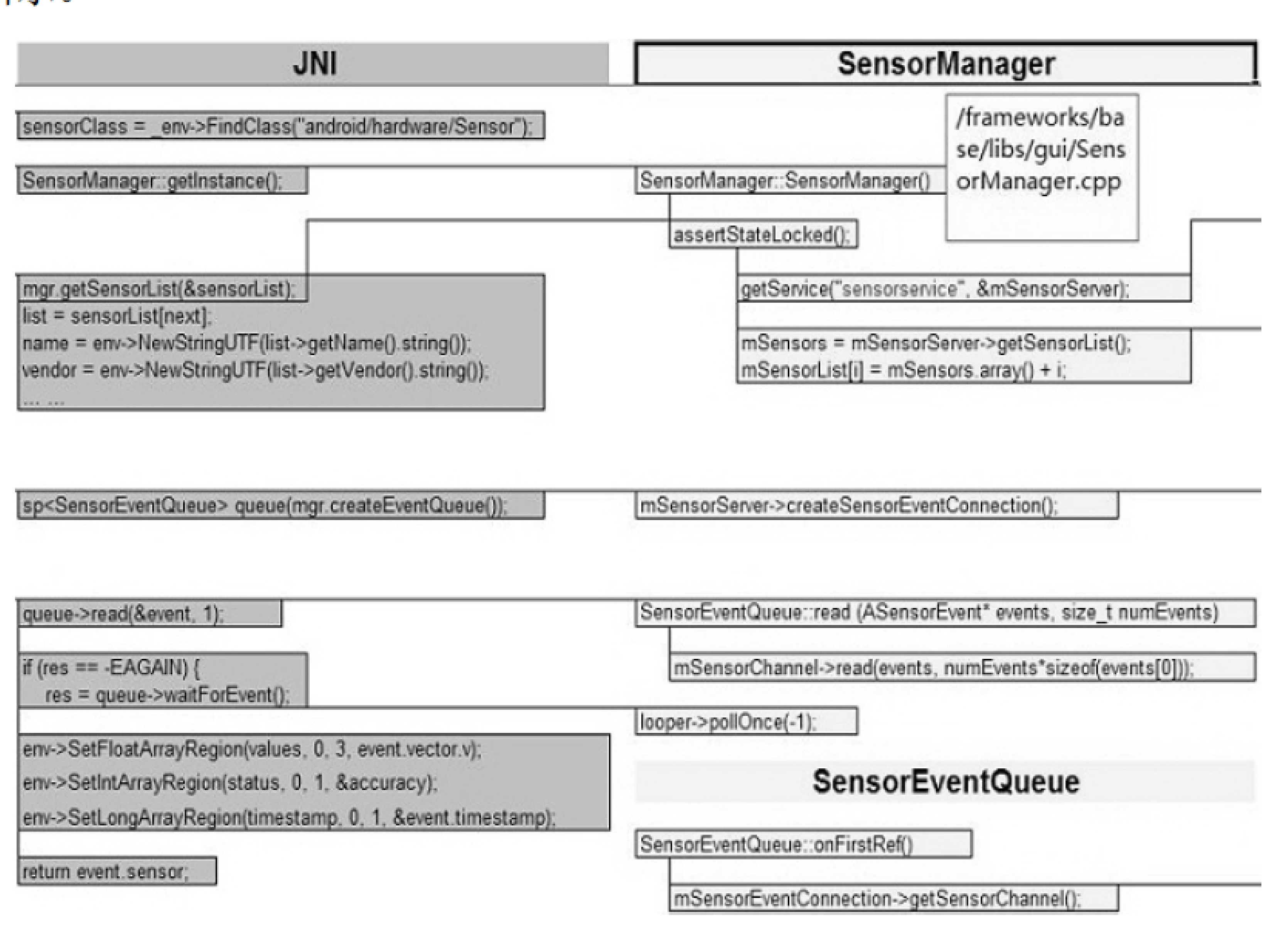


图 5-6　SensorManager 控制流的处理流程

而 SensorEventQueue 负责数据流，功能是通过管道机制来读写底层的数据。具体过程如图 5-7 所示。

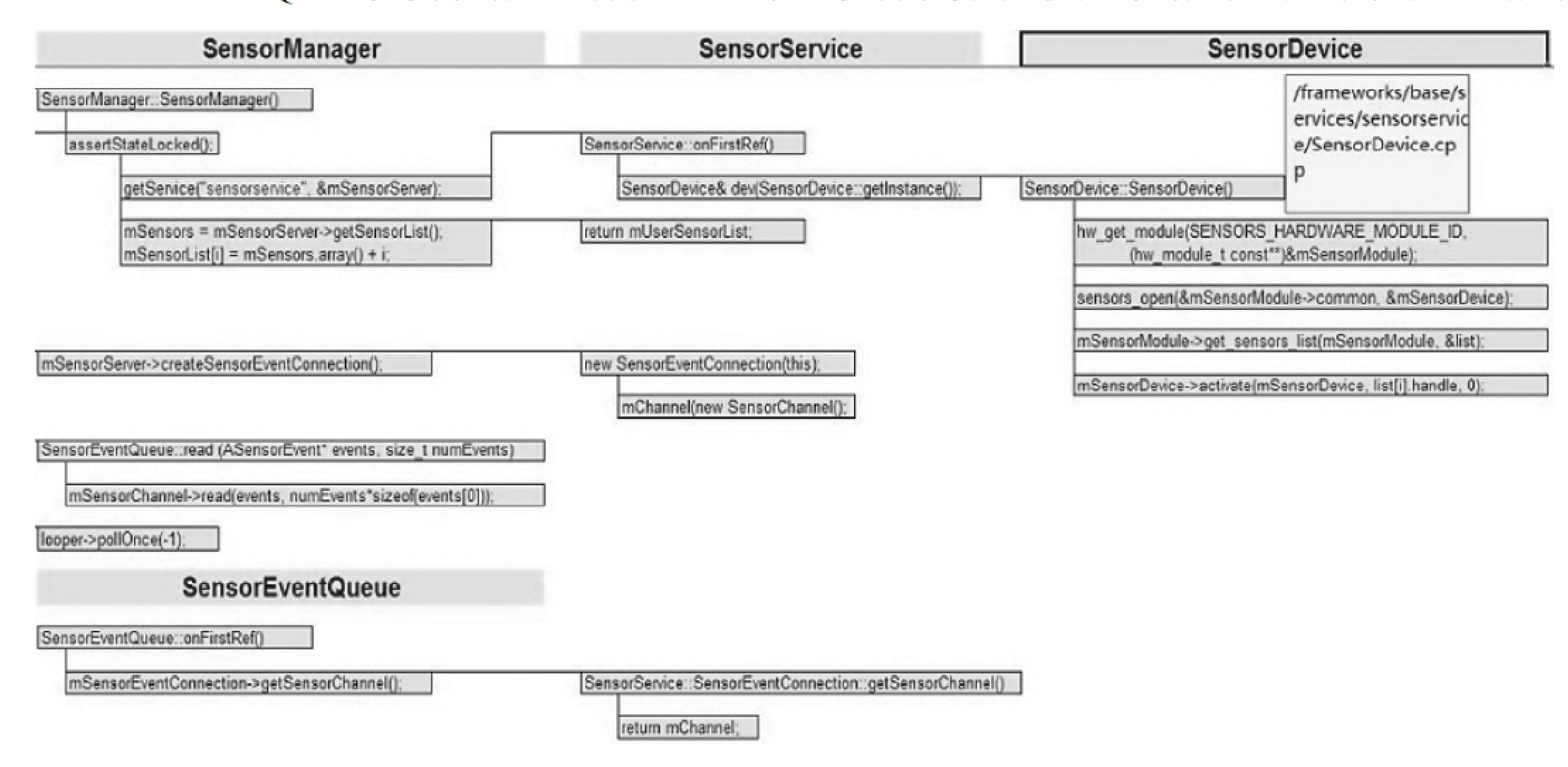


图 5-7　SensorEventQueue 数据流的处理流程

5.1.4　HAL 层详解

在 Android 系统中，HAL 层提供了 Android 独立于具体硬件的抽象接口。其中 HAL 层的头文件路径是：

```
hardware/libhardware/include/hardware/sensors.h
```

而具体实现文件需要开发者个人编写，具体可以参考：

```
hardware\invensense\libsensors_iio\sensors_mpl.cpp
```

文件 sensors.h 的主要实现代码如下所示。

```
typedef struct {
    union {
        float v[3];
        struct {
            float x;
            float y;
            float z;
        };
        struct {
            float azimuth;
            float pitch;
            float roll;
        };
    };
    int8_t status;
    uint8_t reserved[3];
} sensors_vec_t;

/**
 *未校准陀螺仪和磁强计数据事件
 */
typedef struct {
  union {
    float uncalib[3];
    struct {
      float x_uncalib;
      float y_uncalib;
      float z_uncalib;
    };
  };
  union {
    float bias[3];
    struct {
      float x_bias;
      float y_bias;
      float z_bias;
    };
  };
} uncalibrated_event_t;
```

```
/**
 *各种类型的传感器数据中的联合
 *可以返回
 */
typedef struct sensors_event_t {
    int32_t version;
    int32_t sensor;

    /* 传感器类型 */
    int32_t type;
    int32_t reserved0;

    /* 事件微秒 */
    int64_t timestamp;
    union {
        float           data[16];

        sensors_vec_t   acceleration;
        sensors_vec_t   magnetic;
        sensors_vec_t   orientation;
        sensors_vec_t   gyro;
        float           temperature;
        float           distance;

        float           light;

        float           pressure;

        float           relative_humidity;

        uint64_t        step_counter;

        uncalibrated_event_t uncalibrated_gyro;

        uncalibrated_event_t uncalibrated_magnetic;
    };
    uint32_t        reserved1[4];
} sensors_event_t;

struct sensor_t;
struct sensors_module_t {
    struct hw_module_t common;
    int (*get_sensors_list)(struct sensors_module_t* module,
            struct sensor_t const** list);
};

struct sensor_t {
    const char*     name;

    const char*     vendor;
```

```
    int                 version;
    int                 handle;

    int                 type;

    float               maxRange;

    float               resolution;

    float               power;
    int32_t             minDelay;

    void*               reserved[8];
};

struct sensors_poll_device_t {
    struct hw_device_t common;
    int (*activate)(struct sensors_poll_device_t *dev,
            int handle, int enabled);
    int (*setDelay)(struct sensors_poll_device_t *dev,
            int handle, int64_t ns);
    int (*poll)(struct sensors_poll_device_t *dev,
            sensors_event_t* data, int count);
};
typedef struct sensors_poll_device_1 {
    union {
        struct sensors_poll_device_t v0;

        struct {
            struct hw_device_t common;
            int (*activate)(struct sensors_poll_device_t *dev,
                    int handle, int enabled);

            int (*setDelay)(struct sensors_poll_device_t *dev,
                    int handle, int64_t period_ns);

            int (*poll)(struct sensors_poll_device_t *dev,
                    sensors_event_t* data, int count);
        };
    };
    int (*batch)(struct sensors_poll_device_1* dev,
            int handle, int flags, int64_t period_ns, int64_t timeout);

    void (*reserved_procs[8])(void);

} sensors_poll_device_1_t;



/**用于打开和关闭的装置方便的 API */
```

```
static inline int sensors_open(const struct hw_module_t* module,
        struct sensors_poll_device_t** device) {
    return module->methods->open(module,
            SENSORS_HARDWARE_POLL, (struct hw_device_t**)device);
}

static inline int sensors_close(struct sensors_poll_device_t* device) {
    return device->common.close(&device->common);
}

static inline int sensors_open_1(const struct hw_module_t* module,
        sensors_poll_device_1_t** device) {
    return module->methods->open(module,
            SENSORS_HARDWARE_POLL, (struct hw_device_t**)device);
}

static inline int sensors_close_1(sensors_poll_device_1_t* device) {
    return device->common.close(&device->common);
}

__END_DECLS

#endif  // ANDROID_SENSORS_INTERFACE_H
```

而具体的实现文件是 Linux Kernel 层，也就是具体的硬件设备驱动程序，例如可以将其命名为 sensors.c，然后编写如下定义 struct sensors_poll_device_t 的代码。

```
    struct sensors_poll_device_t {
        struct hw_device_t common;

        //激活/停用一个传感器
        int (*activate)(struct sensors_poll_device_t *dev,
                int handle, int enabled);

        //对于一个给定的传感器，设置在微秒传感器事件之间的延迟
        int (*setDelay)(struct sensors_poll_device_t *dev,
                int handle, int64_t ns);

        //返回传感器数据的数组
        int (*poll)(struct sensors_poll_device_t *dev,
                sensors_event_t* data, int count);
    };
```

也可以编写如下定义 struct sensors_module_t 的代码。

```
    struct sensors_module_t {
        struct hw_module_t common;

        /**
         *枚举所有可用的传感器。这份名单是在“名单”返回
         *@传感器在列表中返回数
         */
        int (*get_sensors_list)(struct sensors_module_t* module,
                struct sensor_t const** list);
    };
```

也可以编写如下定义 struct sensor_t 的代码。

```
struct sensor_t {
    const char*     name;
    int             version;
    int             handle;
    int             type;
    float           maxRange;
    float           resolution;
    float           power;
    int32_t         minDelay;
    void*           reserved[8];
};
```

也可以编写如下定义 struct sensors_event_t 的代码。

```
typedef struct {
    union {
        float v[3];
        struct {
            float x;
            float y;
            float z;
        };
        struct {
            float azimuth;
            float pitch;
            float roll;
        };
    };
    int8_t status;
    uint8_t reserved[3];
} sensors_vec_t;
typedef struct sensors_event_t {
    int32_t version;

    int32_t sensor;

    int32_t type;
    int32_t reserved0;

    int64_t timestamp;

    union {
        float           data[16];
        sensors_vec_t   acceleration;

        sensors_vec_t   magnetic;

        sensors_vec_t   orientation;

        sensors_vec_t   gyro;
```

```
                float           temperature;

                float           distance;

                float           light;

                float           pressure;

                float           relative_humidity;
            };
            uint32_t        reserved1[4];
        } sensors_event_t;
```

也可以编写如下定义 struct sensors_module_t 的代码。

```
static const struct sensor_t sSensorList[] = {
        { "MMA8452Q 3-axis Accelerometer",
                "Freescale Semiconductor",
                1, SENSORS_HANDLE_BASE+ID_A,
                SENSOR_TYPE_ACCELEROMETER, 4.0f*9.81f, (4.0f*9.81f)/256.0f, 0.2f, 0, { } },
        { "AK8975 3-axis Magnetic field sensor",
                "Asahi Kasei",
                1, SENSORS_HANDLE_BASE+ID_M,
                SENSOR_TYPE_MAGNETIC_FIELD, 2000.0f, 1.0f/16.0f, 6.8f, 0, { } },
        { "AK8975 Orientation sensor",
                "Asahi Kasei",
                1, SENSORS_HANDLE_BASE+ID_O,
                SENSOR_TYPE_ORIENTATION, 360.0f, 1.0f, 7.0f, 0, { } },

    { "ST 3-axis Gyroscope sensor",
            "STMicroelectronics",
            1, SENSORS_HANDLE_BASE+ID_GY,
            SENSOR_TYPE_GYROSCOPE, RANGE_GYRO, CONVERT_GYRO, 6.1f, 1190, { } },

    { "AL3006Proximity sensor",
        "Dyna Image Corporation",
        1, SENSORS_HANDLE_BASE+ID_P,
        SENSOR_TYPE_PROXIMITY,
        PROXIMITY_THRESHOLD_CM, PROXIMITY_THRESHOLD_CM,
        0.5f, 0, { } },

        { "AL3006 light sensor",
                "Dyna Image Corporation",
                1, SENSORS_HANDLE_BASE+ID_L,
                SENSOR_TYPE_LIGHT, 10240.0f, 1.0f, 0.5f, 0, { } },

};

static int open_sensors(const struct hw_module_t* module, const char* name,
        struct hw_device_t** device);

static int sensors__get_sensors_list(struct sensors_module_t* module,
        struct sensor_t const** list)
{
```

```
    *list = sSensorList;
    return ARRAY_SIZE(sSensorList);
}

static struct hw_module_methods_t sensors_module_methods = {
    .open = open_sensors
};

const struct sensors_module_t HAL_MODULE_INFO_SYM = {
    .common = {
        .tag = HARDWARE_MODULE_TAG,
        .version_major = 1,
        .version_minor = 0,
        .id = SENSORS_HARDWARE_MODULE_ID,
        .name = "MMA8451Q & AK8973A & gyro Sensors Module",
        .author = "The Android Project",
        .methods = &sensors_module_methods,
    },
    .get_sensors_list = sensors__get_sensors_list
};

static int open_sensors(const struct hw_module_t* module, const char* name,
        struct hw_device_t** device)
{
    return init_nusensors(module, device); //待后面讲解
}
```

到此为止，整个 Android 系统中传感器模块的源码分析完毕。由此可见，整个传感器系统的总体调用关系如图 5-8 所示。

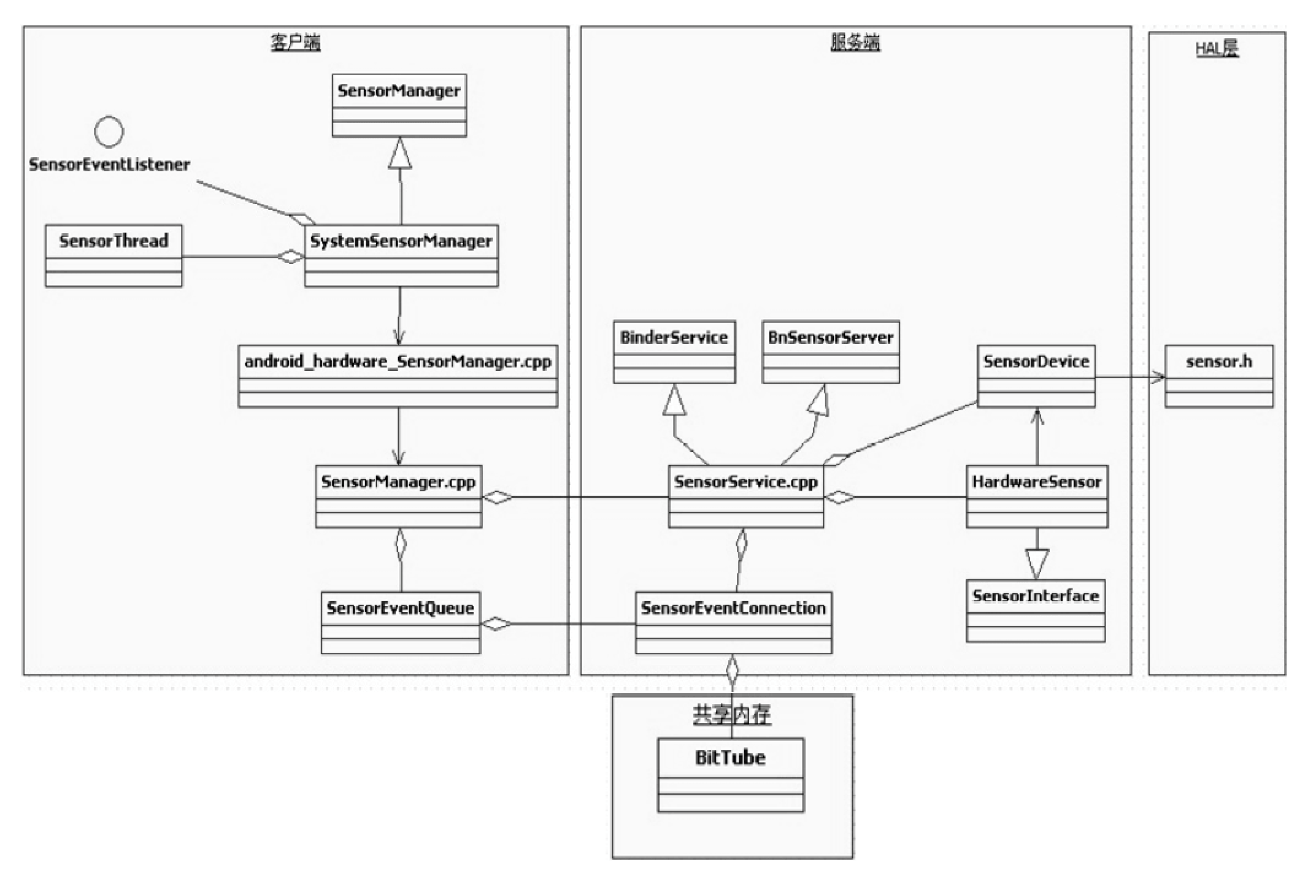


图 5-8　传感器系统的总体调用关系

客户端读取数据时的调用时序如图 5-9 所示。

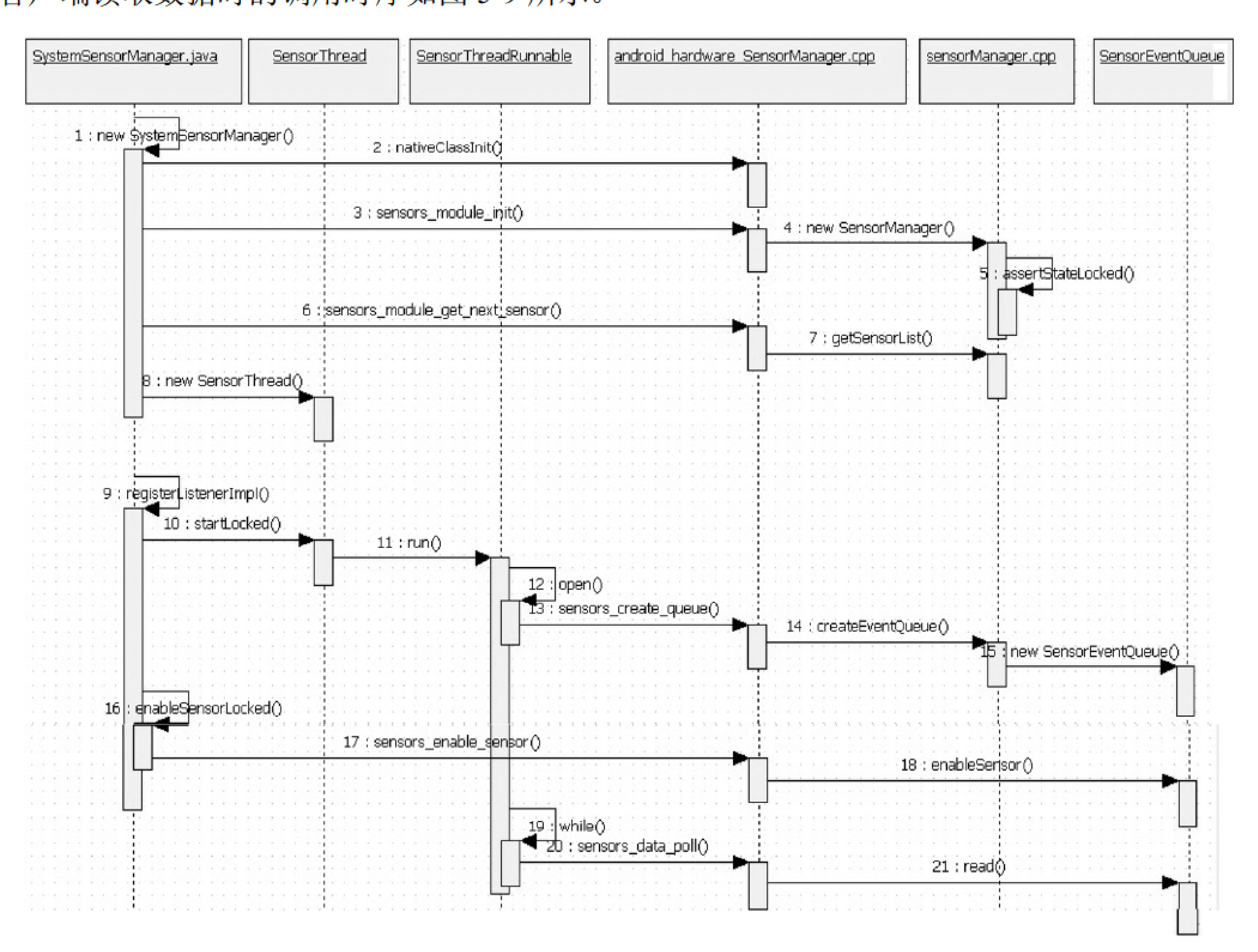


图 5-9　客户端读取数据时的调用时序图

服务器端的调用时序如图 5-10 所示。

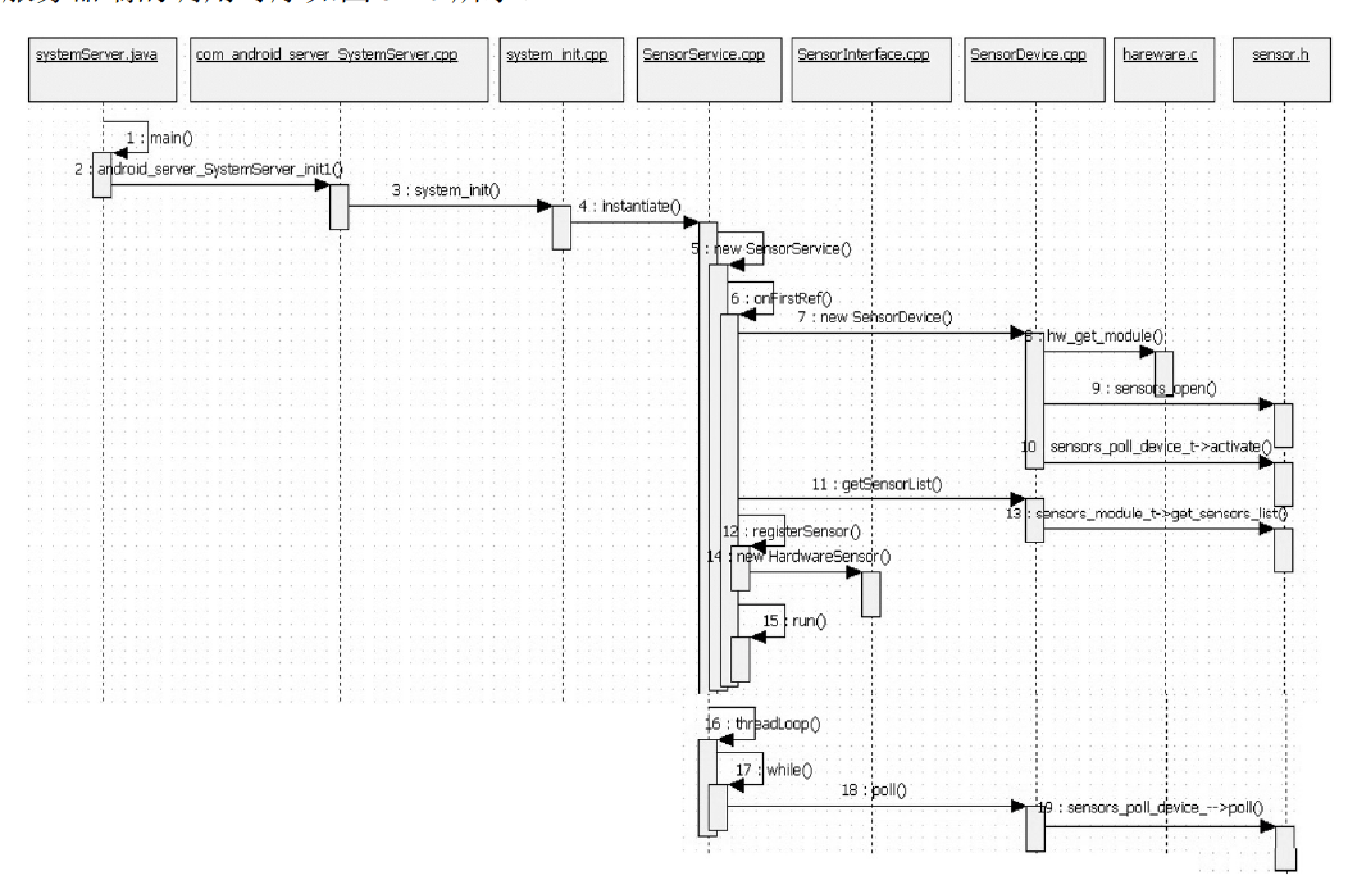
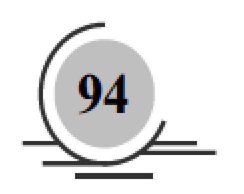


图 5-10　服务器端的调用时序图



5.2 Android 传感器应用开发基础

知识点讲解：光盘:视频\知识点\第 5 章\Android 传感器应用开发基础.avi

在本章前面的内容中，已经详细讲解了 Android 系统中传感器系统的架构知识。在现实应用中，传感器系统在外设项目开发过程、可穿戴设备和家居设备中得到了广泛的应用。本节将详细讲解开发 Android 传感器应用程序的基础知识，介绍使用传感器技术开发外设项目开发过程应用程序的基本流程，为读者学习本书后面的知识打下坚实的基础。

5.2.1 查看包含的传感器

在安装 Android SDK 后，依次打开安装目录中的如下帮助文件：

```
android SDK/sdk/docs/reference/android/hardware/Sensor.html
```

在此文件中列出了 Android 传感器系统所包含的所有传感器类型，如图 5-11 所示。

Android APIs　API level: 18

android.gesture
android.graphics
android.graphics.drawable
android.graphics.drawable.shapes
android.hardware
android.hardware.display
android.hardware.input
android.hardware.location
android.hardware.usb
android.inputmethodservice
android.location
android.media
android.media.audiofx
android.media.effect

Camera.Area
Camera.CameraInfo
Camera.Face
Camera.Parameters
Camera.Size
GeomagneticField
Sensor
SensorEvent
SensorManager
TriggerEvent
TriggerEventListener

Use Tree Navigation

Summary

Constants		
int	TYPE_ACCELEROMETER	A constant describing an accelerometer sensor type.
int	TYPE_ALL	A constant describing all sensor types.
int	TYPE_AMBIENT_TEMPERATURE	A constant describing an ambient temperature sensor type
int	TYPE_GAME_ROTATION_VECTOR	Identical to TYPE_ROTATION_VECTOR except that it doesn't use the geomagnetic field.
int	TYPE_GRAVITY	A constant describing a gravity sensor type.
int	TYPE_GYROSCOPE	A constant describing a gyroscope sensor type
int	TYPE_GYROSCOPE_UNCALIBRATED	A constant describing a gyroscope uncalibrated sensor type.
int	TYPE_LIGHT	A constant describing a light sensor type.
int	TYPE_LINEAR_ACCELERATION	A constant describing a linear acceleration sensor type.
int	TYPE_MAGNETIC_FIELD	A constant describing a magnetic field sensor type.
int	TYPE_MAGNETIC_FIELD_UNCALIBRATED	A constant describing a magnetic field uncalibrated sensor type.
int	TYPE_ORIENTATION	*This constant was deprecated in API level 8. use* SensorManager.getOrientation() *instead.*
int	TYPE_PRESSURE	A constant describing a pressure sensor type
int	TYPE_PROXIMITY	A constant describing a proximity sensor type.
int	TYPE_RELATIVE_HUMIDITY	A constant describing a relative humidity sensor type.
int	TYPE_ROTATION_VECTOR	A constant describing a rotation vector sensor type.
int	TYPE_SIGNIFICANT_MOTION	A constant describing the significant motion trigger sensor.
int	TYPE_TEMPERATURE	*This constant was deprecated in API level 14. use* Sensor.TYPE_AMBIENT_TEMPERATURE *instead.*

图 5-11 Android 传感器系统的类型

另外，也可以直接在线登录 http://developer.android.com/reference/android/hardware/Sensor.html 来查看。由此可见，在当前最新（作者写稿时最新）版本 Android 4.4 中一共提供了 18 种传感器 API。各个类型的具体说明如下。

（1）TYPE_ACCELEROMETER：加速度传感器，单位是 m/s^2，测量应用于设备 X、Y、Z 轴上的加速度，又叫做 G-sensor。

（2）TYPE_AMBIENT_TEMPERATURE：温度传感器，单位是℃，能够测量并返回当前的温度。

（3）TYPE_GRAVITY：重力传感器，单位是 m/s^2，用于测量设备 X、Y、Z 轴上的重力，也叫 GV-sensor，地球上的数值是 $9.8m/s^2$，也可以设置其他星球。

（4）TYPE_GYROSCOPE：陀螺仪传感器，单位是 rad/s，能够测量设备 X、Y、Z 三轴的角加速

度数据。

（5）TYPE_LIGHT：光线感应传感器，单位是 lx，能够检测周围的光线强度，在手机系统中主要用于调节 LCD 亮度。

（6）TYPE_LINEAR_ACCELERATION：线性加速度传感器，单位是 m/s^2，能够获取加速度传感器去除重力的影响得到的数据。

（7）TYPE_MAGNETIC_FIELD：磁场传感器，单位是 μT（微特斯拉），能够测量设备周围 3 个物理轴（X，Y，Z）的磁场。

（8）TYPE_ORIENTATION：方向传感器，用于测量设备围绕 3 个物理轴（X，Y，Z）的旋转角度，在新版本中已经使用 SensorManager.getOrientation()替代。

（9）TYPE_PRESSURE：气压传感器，单位是 hPa（百帕斯卡），能够返回当前环境下的压强。

（10）TYPE_PROXIMITY：距离传感器，单位是 cm，能够测量某个对象到屏幕的距离。可以在打电话时判断人耳到电话屏幕的距离，以关闭屏幕而达到省电功能。

（11）TYPE_RELATIVE_HUMIDITY：湿度传感器，单位是%，能够测量周围环境的相对湿度。

（12）TYPE_ROTATION_VECTOR：旋转向量传感器，旋转矢量代表设备的方向，是一个将坐标轴和角度混合计算得到的数据。

（13）TYPE_TEMPERATURE：温度传感器，在新版本中被 TYPE_AMBIENT_TEMPERATURE 替换。

（14）TYPE_ALL：返回所有的传感器类型。

（15）TYPE_GAME_ROTATION_VECTOR：除了不能使用地磁场之外，和 TYPE_ROTATION_VECTOR 的功能完全相同。

（16）TYPE_GYROSCOPE_UNCALIBRATED：提供了能够让应用调整传感器的原始值，定义了一个描述未校准陀螺仪的传感器类型。

（17）TYPE_MAGNETIC_FIELD_UNCALIBRATED：和 TYPE_GYROSCOPE_UNCALIBRATED 相似，也提供了能够让应用调整传感器的原始值，定义了一个描述未校准陀螺仪的传感器类型。

（18）TYPE_SIGNIFICANT_MOTION：运动触发传感器，应用程序不需要为这种传感器触发任何唤醒锁。能够检测当前设备是否运动，并发送检测结果。

5.2.2 模拟器测试工具——SensorSimulator

在进行和传感器相关的开发工作时，使用 SensorSimulator 测试工具可以提高开发效率。测试工具 SensorSimulator 是一个开源免费的传感器工具，通过该工具可以在模拟器中调试传感器的应用。搭建 SensorSimulator 开发环境的基本流程如下。

（1）下载 SensorSimulator，读者可从 http://code.google.com/p/openintents/wiki/SensorSimulator 网站找到该工具的下载链接。笔者下载的是 sensorsimulator-1.1.1.zip 版本，如图 5-12 所示。

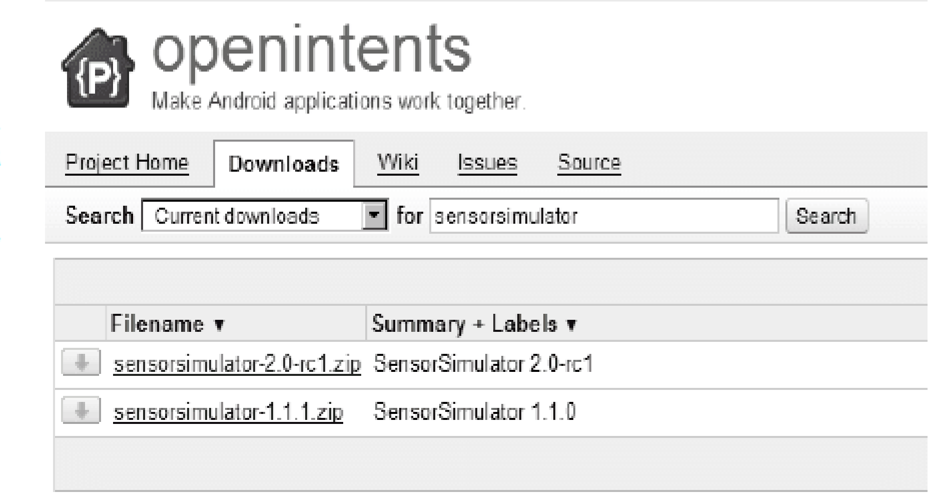
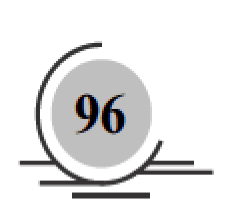


图 5-12 下载 sensorsimulator-1.1.1.zip

（2）将下载好的 SensorSimulator 解压到本地根目录，例如 C 盘的根目录。

（3）向模拟器安装 sensorsimulatorsettings-1.1.1.apk。首先



在操作系统中依次选择“开始” | “运行”命令进入“运行”对话框。

（4）在“运行”对话框中输入 cmd 进入 cmd 命令行，之后通过 cd 命令将当前目录导航到 sensorsimulatorsettings-1.1.1.apk 目录下，然后输入下列命令向模拟器安装该 apk。

```
adb install sensorsimulatorsettings-1.1.1.apk
```

在此需要注意的是，安装 apk 时，一定要保证模拟器正在运行才可以，安装成功后会输出 Success 提示，如图 5-13 所示。

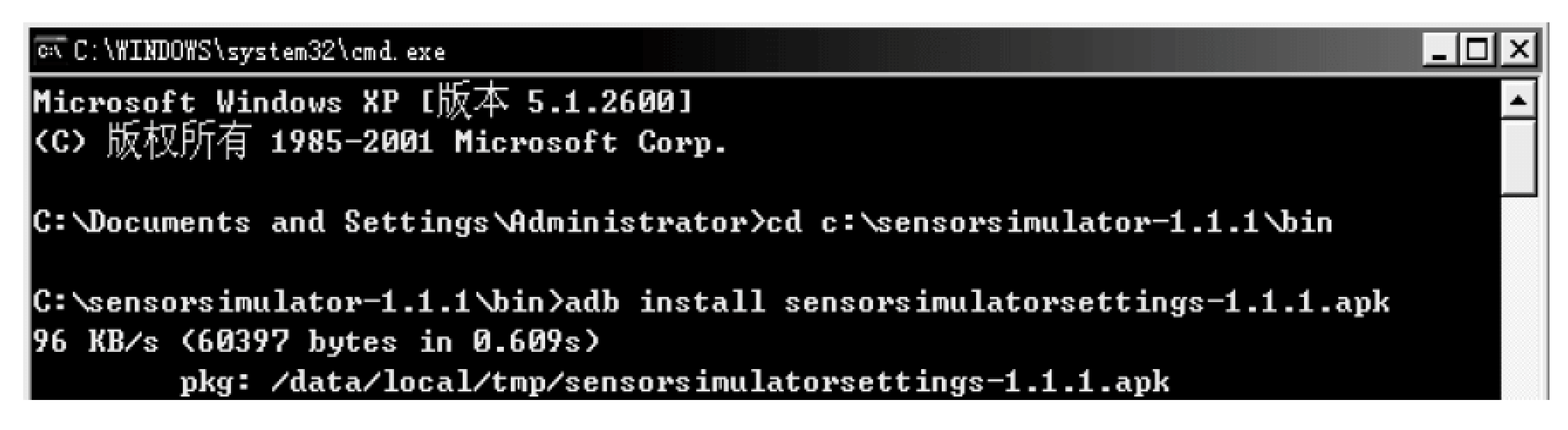


图 5-13　安装 apk

接下来开始配置应用程序，假设我们要在项目 jiaSCH 中使用 SensorSimulator，则配置流程如下。

（1）在 Eclipse 中打开项目 jiaSCH，然后为该项目添加 JAR 包，使其能够使用 SensorSimulator 工具的类和方法。添加方法非常简单，在 Eclipse 的 Package Explorer 中找到该项目的文件夹 jiaSCH，然后右键单击该文件夹，在弹出的快捷菜单中选择 Properties 命令，弹出如图 5-14 所示的 Properties for jiaS 对话框。

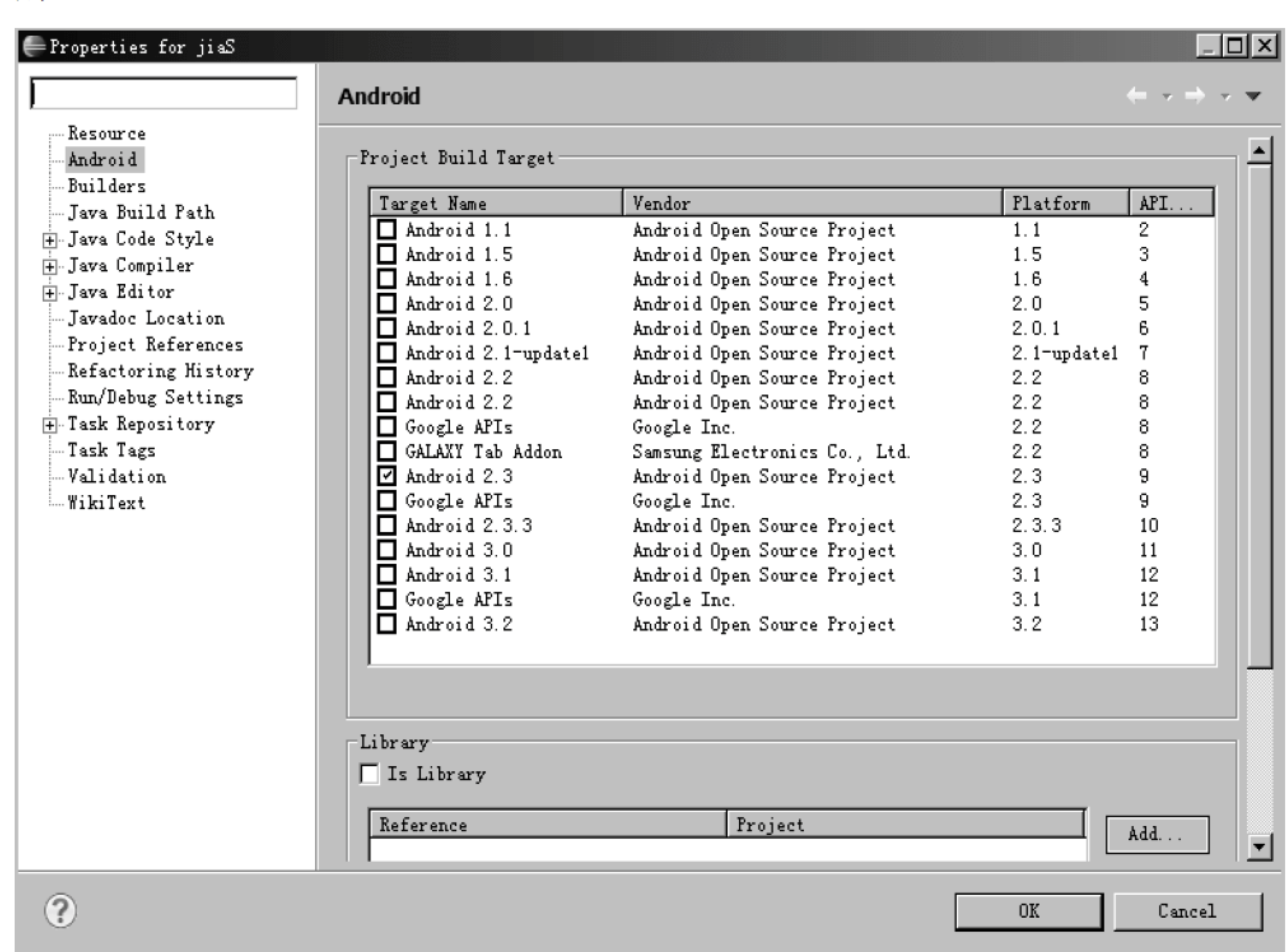


图 5-14　Properties for jiaS 对话框

（2）选择左面的 Java Build Path 选项，然后选择 Libraries 选项卡，如图 5-15 所示。

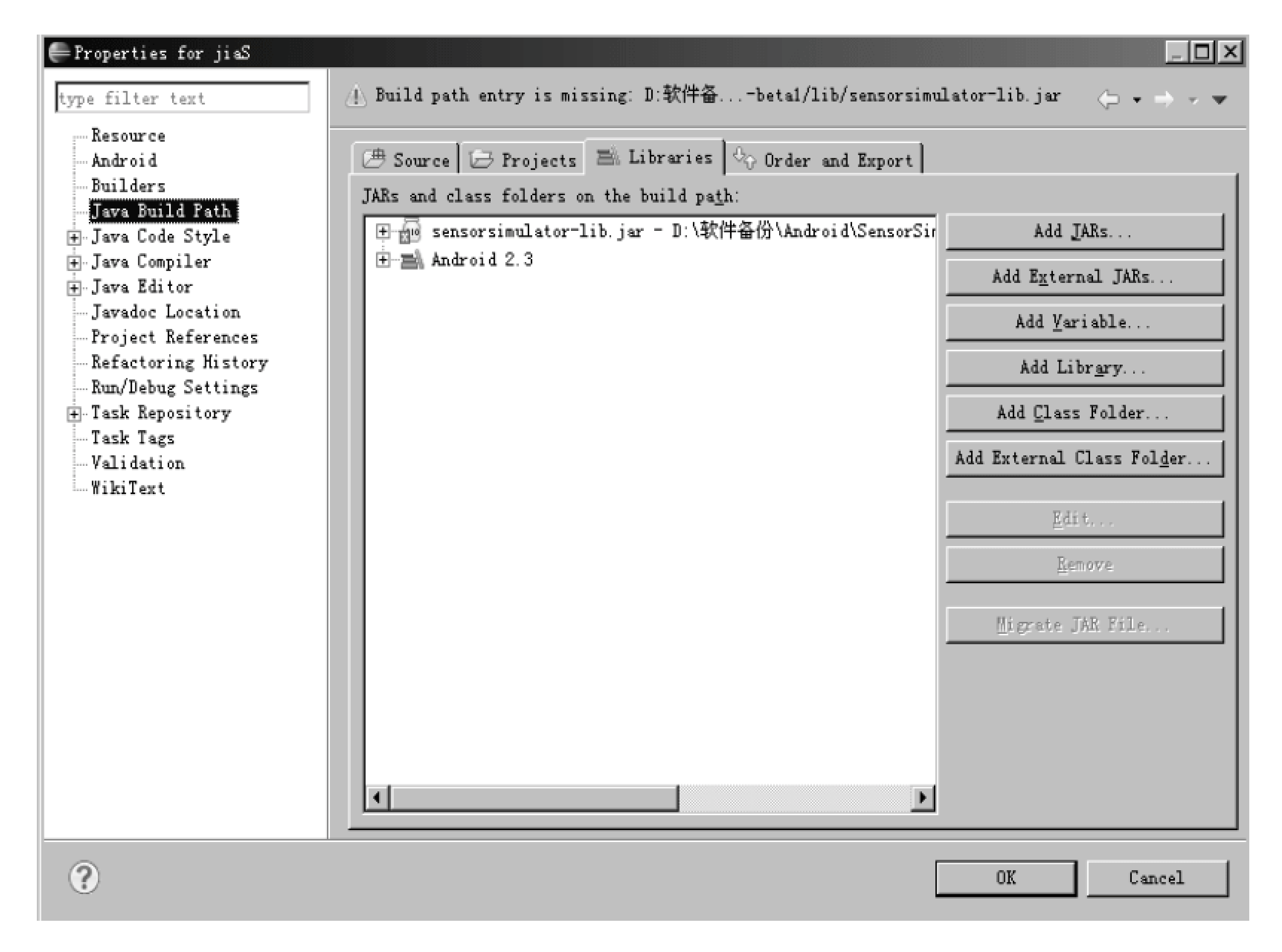


图 5-15　Libraries 选项卡

（3）单击 Add External JARs 按钮，在弹出的 JAR Selection 对话框中找到 sensorsimulator 安装目录下的 sensorsimulator-lib-1.1.1.jar，并将其添加到该项目中，如图 5-16 所示。

（4）开始启动 sensorsimulator.jar，并对手机模拟器上的 SensorSimulatorSettings 进行必要的配置。首先在 C:\sensorsimulator-1.1.1\bin 目录下找到 sensorsimulator.jar 并启动，运行后的界面如图 5-17 所示。

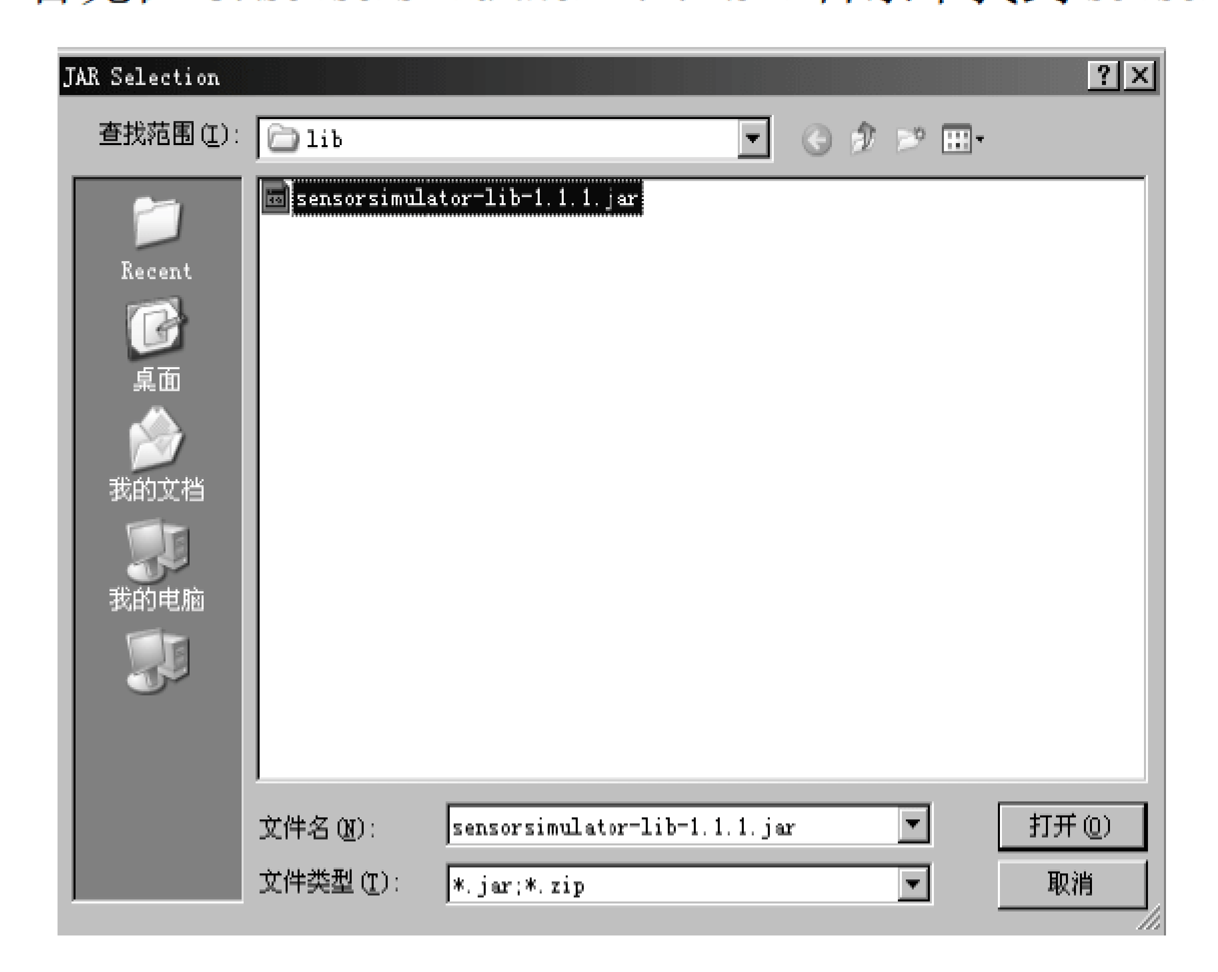


图 5-16　添加需要的 JAR 包

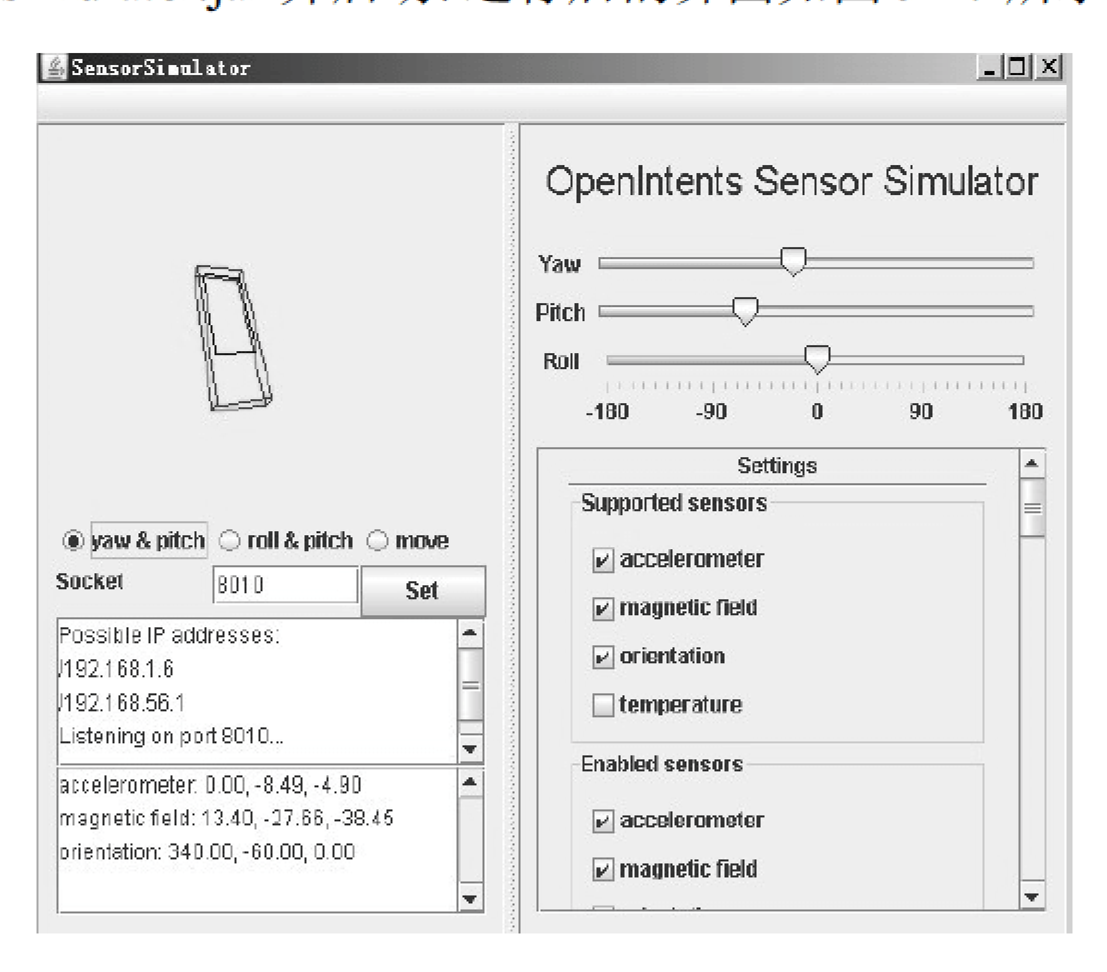
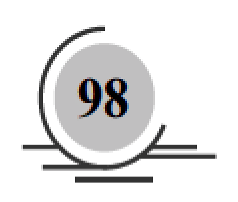


图 5-17　传感器的模拟器

（5）接下来开始进行手机模拟器和 SensorSimulator 的连接配置工作，运行手机模拟器上安装好的 sensorsimulatorsettings-1.1.1.apk，如图 5-18 所示。

（6）在图 5-18 中输入 SensorSimulator 启动时显示的 IP 地址和端口号，单击屏幕右上角的 Testing 按钮后会来到测试连接界面，如图 5-19 所示。

（7）单击屏幕上的 Connect 按钮进入下一界面，如图 5-20 所示。在此界面中可以选择需要监听的传感器，如果能够从传感器中读取到数据，说明 SensorSimulator 与手机模拟器连接成功，可以测试自己开发的应用程序了。







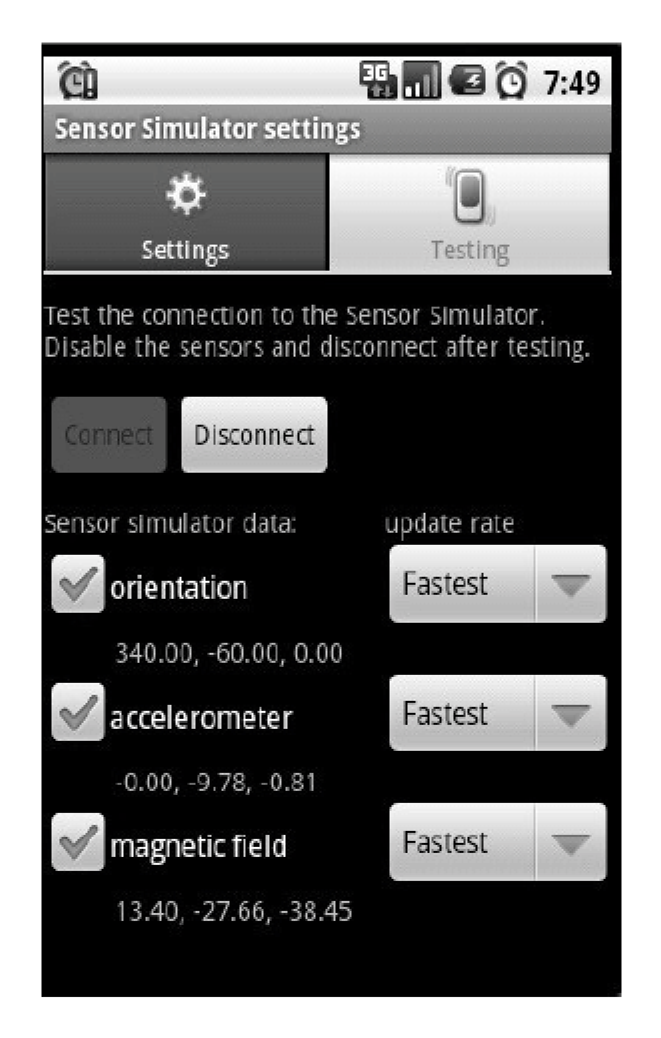


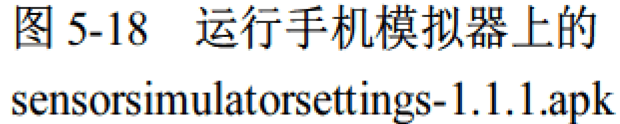
图 5-18　运行手机模拟器上的

sensorsimulatorsettings-1.1.1.apk

图 5-19　测试连接界面

图 5-20　连接界面

到此为止，使用 Eclipse 结合 SensorSimulator 配置传感器应用程序的基本流程介绍完毕。

5.2.3　实战演练——检测当前设备支持的传感器

在接下来的实例中，将演示在 Android 设备中检测当前设备支持传感器类型的方法。

实例	功能	源码路径
实例 5-1	检测当前设备支持的传感器	光盘:\daima\5\SensorEX

本实例的功能是检测当前设备支持的传感器类型，具体实现流程如下。

（1）布局文件 main.xml 的具体实现代码如下所示。

```
<linearlayout android:layout_height="fill_parent" android:layout_width="fill_parent" android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android">
<textview android:layout_height="wrap_content"
 android:layout_width="fill_parent" android:text=""
 android:id="@+id/TextView01"
>
</textview>
</linearlayout>
```

（2）主程序文件 MainActivity.java 的具体实现代码如下所示。

```
public class MainActivity extends Activity {

        /** Called when the activity is first created. */
        @SuppressWarnings("deprecation")
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);

                //准备显示信息的 UI 组建
                final TextView tx1 = (TextView) findViewById(R.id.TextView01);
```

```
                //从系统服务中获得传感器管理器
                SensorManager sm = (SensorManager) getSystemService(Context.SENSOR_SERVICE);

                //从传感器管理器中获得全部的传感器列表
                List<Sensor> allSensors = sm.getSensorList(Sensor.TYPE_ALL);

                //显示有多少个传感器
                tx1.setText("经检测该手机有" + allSensors.size() + "个传感器，它们分别是：\n");

                //显示每个传感器的具体信息
                for (Sensor s : allSensors) {

                        String tempString = "\n" + "  设备名称：" + s.getName() + "\n" + "  设备版本：" +
s.getVersion() + "\n" + "  供应商："
                                + s.getVendor() + "\n";

                        switch (s.getType()) {
                        case Sensor.TYPE_ACCELEROMETER:
                                tx1.setText(tx1.getText().toString() + s.getType() + " 加速度传感器
accelerometer" + tempString);
                                break;
                        case Sensor.TYPE_GYROSCOPE:
                                tx1.setText(tx1.getText().toString() + s.getType() + " 陀螺仪传感器 gyroscope"
+ tempString);
                                break;
                        case Sensor.TYPE_LIGHT:
                                tx1.setText(tx1.getText().toString() + s.getType() + " 环境光线传感器 light"
+ tempString);
                                break;
                        case Sensor.TYPE_MAGNETIC_FIELD:
                                tx1.setText(tx1.getText().toString() + s.getType() + " 电磁场传感器 magnetic field"
+ tempString);
                                break;
                        case Sensor.TYPE_ORIENTATION:
                                tx1.setText(tx1.getText().toString() + s.getType() + " 方向传感器 orientation" +
tempString);
                                break;
                        case Sensor.TYPE_PRESSURE:
                                tx1.setText(tx1.getText().toString() + s.getType() + " 压力传感器 pressure"
+ tempString);
                                break;
                        case Sensor.TYPE_PROXIMITY:
                                tx1.setText(tx1.getText().toString() + s.getType() + " 距离传感器 proximity" +
tempString);
                                break;
                        case Sensor.TYPE_AMBIENT_TEMPERATURE :
                                tx1.setText(tx1.getText().toString() + s.getType() + " 温度传感器 temperature" +
tempString);
                                break;
                        default:
```



```
                                        tx1.setText(tx1.getText().toString() + s.getType() + " 未知传感器" +
tempString);
                                        break;
                              }
                    }

          }
}
```

上述实例代码需要在真机中运行，执行后将会列表显示当前设备所支持的传感器类型，如图 5-21 所示。



图 5-21　执行效果

5.3　光线传感器基础

知识点讲解：光盘:视频\知识点\第 5 章\光线传感器基础.avi

在实际应用中，光线传感器能够根据手机所处环境的光线来调节手机屏幕的亮度和键盘灯。例如在光线充足的地方屏幕会很亮，键盘灯就会关闭。相反，如果在暗处，键盘灯就会亮，屏幕较暗（与屏幕亮度的设置也有关系），这样既保护了眼睛，又节省了能量。光线传感器在进入睡眠模式时会发出蓝色周期性闪动的光，非常美观。本节将详细讲解 Android 光线传感器的基本知识。

5.3.1　光线传感器介绍

在外设项目开发过程中，光线传感器通常位于前摄像头旁边的一个小点，如果在光线充足的情况（室外或者是灯光充足的室内）下，在 2～3 秒之后键盘灯会自动熄灭，即使再操作机器键盘灯也不会亮，除非到了光线比较暗的地方才会自动亮起来。如果在光线充足的情况下用手将光线感应器遮上，在 2～3 秒后键盘灯会自动亮起来，在此过程中光线感应器起到了一个节电的功能。

要想在 Android 外设项目开发过程中监听光线传感器，需要掌握如下监听方法。

（1）registerListener(SensorListenerlistener,int sensors,int rate)：已过时。

（2）registerListener(SensorListenerlistener,int sensors)：已过时。

（3）registerListener(SensorEventListenerlistener,Sensor sensors,int rate)：监听光线变化。

（4）registerListener(SensorEventListenerlistener,Sensor sensors,int rate,Handlerhandler)：因为 SensorListener 已经过时，所以相应的注册方法也过时了。

在上述方法中，各个参数的具体说明如下所示。

- Listener：相应监听器的引用；
- Sensor：相应的感应器引用；
- Rate：感应器的反应速度，这个必须是系统提供的4个常量之一。
  - SENSOR_DELAY_NORMAL：匹配屏幕方向的变化。
  - SENSOR_DELAY_UI：匹配用户接口。
  - SENSOR_DELAY_GAME：匹配游戏。
  - SENSOR_DELAY_FASTEST.：匹配所能达到的最快。

开发光线传感器应用时需要监测 SENSOR_LIGHT，例如下面的代码。

```
private SensorListener mySensorListener = new SensorListener(){
@Override
 public void onAccuracyChanged(int sensor, int accuracy) { }       //重写 onAccuracyChanged()方法
 @Override
 public void onSensorChanged(int sensor, float[ ] values) {        //重写 onSensorChanged()方法
     if(sensor == SensorManager.SENSOR_LIGHT){                     //只检查光强度的变化
         myTextView1.setText("光的强度为："+values[0]);              //将光的强度显示到 TextView
             }
        }
};
@Override
protected void onResume() {                                        //重写的 onResume()方法
         mySensorManager.registerListener(                         //注册监听
                 mySensorListener,                                 //监听器 SensorListener 对象
                 SensorManager.SENSOR_LIGHT,                       //传感器的类型为光的强度
                 SensorManager.SENSOR_DELAY_UI                     //频率
                 );
         super.onResume();
}
```

在上述代码中，通过 if 语句判断是否为光的强度改变事件。在代码中只对光强度改变事件进行处理，将得到的光强度显示在屏幕中。光线传感器只得到一个数据，而并不像其他传感器那样得到的是 X、Y、Z 三个方向上的分量。

在注册监听时，通过传入 SensorManager.SENSOR_LIGHT 来通知系统只注册光线传感器。

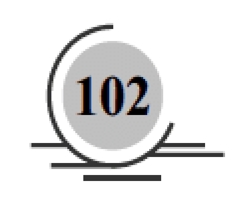
5.3.2 使用光线传感器的方法

在 Android 外设项目开发过程中，使用光线传感器的基本流程如下。

（1）通过一个 SensorManager 来管理各种感应器，要想获得这个管理器的引用，必须通过如下所示的代码来实现。

```
(SensorManager)getSystemService(Context.SENSOR_SERVICE);
```

（2）在 Android 系统中，所有的感应器都属于 Sensor 类的一个实例，并没有继续细分下去，所以 Android 对于感应器的处理几乎是一模一样的。既然都是 Sensor 类，那么怎么获得相应的感应器呢？



这时就需要通过 SensorManager 来获得，可以通过如下所示的代码来确定要获得的感应器类型。

```
sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

通过上述代码获得了光线感应器的引用。

（3）在获得相应的传感器的引用后可以来感应光线强度的变化，此时需要通过监听传感器的方式来获得变化，监听功能通过前面介绍的监听方法实现。Android 提供了两个监听方式，一个是 SensorEventListener，另一个是 SensorListener，后者已经在 Android API 上显示过时了。

（4）在 Android 中注册传感器后，此时就说明启用了传感器。使用感应器是相当耗电的，这也是为什么传感器的应用没有那么广泛的主要原因，所以必须在不需要的时候及时关掉。在 Android 中通过如下注销方法来关闭。

- unregisterListener(SensorEventListenerlistener);
- unregisterListener(SensorEventListenerlistener,Sensor sensor);

（5）使用 SensorEventListener 来具体实现，在 Android 外设项目开发过程中有如下两种实现这个监听器的方法。

- onAccuracyChanged(Sensor sensor, int accuracy)：是反映速度变化的方法，也就是rate变化时的方法。
- onSensorChanged(SensorEvent event)：是传感器的值变化的相应的方法。

读者需要注意的是，上述两个方法会同时响应。也就是说，当感应器发生变化时，这两个方法会一起被调用。上述方法中的 accuracy 的值是 4 个常量，对应的整数如下。

- SENSOR_DELAY_NORMAL：3。
- SENSOR_DELAY_UI：2。
- SENSOR_DELAY_GAME：1。
- SENSOR_DELAY_FASTEST：0。

而类 SensorEvent 有 4 个成员变量，具体说明如下。

- Accuracy：精确值。
- Sensor：发生变化的感应器。
- Timestamp：发生的时间，单位是纳秒。
- Values：发生变化后的值，这个是一个长度为3的数组。

光线传感器只需要 values[0]的值，其他两个都为 0，而 values[0]就是开发光线传感器所需要的，单位是 lux 照度单位。

5.4　磁场传感器详解

知识点讲解：光盘:视频\知识点\第 5 章\磁场传感器详解.avi

在实际应用中，经常需要检测 Android 设备的方向，例如设备的朝向和移动方向。在 Android 系统中，通常使用重力传感器、加速度传感器、磁场传感器和旋转矢量传感器来检测设备的方向。本节将详细讲解在 Android 设备中使用磁场传感器检测设备方向的基本知识。

5.4.1 什么是磁场传感器

磁场传感器是可以将各种磁场及其变化的量转变成电信号输出的装置。自然界和人类社会生活的许多地方都存在磁场或与磁场相关的信息。 磁场传感器是利用人工设置的永久磁体产生的磁场，可以作为许多种信息的载体，被广泛用于探测、采集、存储、转换、复现和监控各种磁场和磁场中承载的各种信息的任务。在当今的信息社会中，磁场传感器已成为信息技术和信息产业中不可缺少的基础元件。目前，人们已研制出利用各种物理、化学和生物效应的磁场传感器，并已在科研、生产和社会生活的各个方面得到广泛应用，承担起探究种种信息的任务。

在现实市面中，最早磁场传感器是伴随测磁仪器的进步而逐步发展的。在众多的测试磁场方法中，大多都是将磁场信息变成电信号进行测量。在测磁仪器中“探头”或“取样装置”就是磁场传感器。随着信息产业、工业自动化、交通运输、电力电子技术、办公自动化、家用电器、医疗仪器等的飞速发展和电子计算机应用的普及，需用大量的传感器将需进行测量和控制的非电参量，转换成可与计算机兼容的信号，作为它们的输入信号，这就给磁场传感器的快速发展提供了机会，形成了相当可观的磁场传感器产业。

5.4.2 磁场传感器的分类

在现实应用中，磁场传感器的主要分类如下。

（1）薄膜磁致电阻传感器

铁磁性物质在磁化过程中，它的电阻值沿磁化方向将增加，并达到饱和的现象称为磁阻效应。薄膜磁阻元件是利用薄膜工艺和微细加工技术，将 NiPe/NiCo 合金用真空蒸镇或溯射工艺沉积到硅片或铁氧体基片上，通过微细加工技术制成一定形状的磁咀图形，形成三端式、四端式以及多端式器件。BMber 结构桥式电路磁阻元件具有灵敏度高、工作频率特性好、温度稳定性好、结构简单、体积小等特点，可制成高密度磁咀磁头、磁性编码器、磁阻位移传感器、磁阻电流传感器等。

（2）磁阻敏感器

物质在磁场中电阻发生变化的现象称为磁阻效应。对于铁、钻、镍及其合金等强磁性金属，当外加磁场平行于磁体内部磁化方向时，电阻几乎不随外加磁场变化；当外加磁场偏离金属的内磁化方向时，此类金属的电阻值将减小，这就是强磁金属的各向异性磁阻效应。

（3）电涡流式传感器

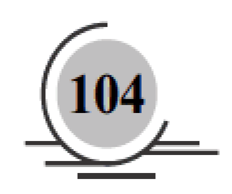
近年来，国内外正发展一对建立在电涡流效应原理上的传感器，即电涡流式传感器。这种传感器不但具有测量线性范围大、灵敏度高、结构简单、抗干扰能力强、不受油污等介质的影响等优点，而且又具有无损、非接触测量的特点，目前正广泛地应用于工业各部门中的位移、尺寸、厚度、振动、转速、压力、电导率、温度、波面等测量，以及探测金属材料和加工件表面裂纹及缺陷。

（4）磁性液体加速度传感器

磁性液体作为一种新型的纳米功能材料，一经问世便走到科学技术发展的前沿，目前科学家们已经将这种新型功能材料应用到广阔的领域中。以此为基础的磁性液体传感器技术也引起了国际技术领域广泛的关注。

（5）磁性液体水平传感器

磁性液体传感器的研究与应用起源于美国。早在 1983 年美国新墨西哥州阿尔帕克基应用技术公司



就与美国空军签订了合同，美国新墨西哥州阿尔帕克基应用技术公司便开始研制基于磁性液体动力学原理（MHD）的主动式和被动式传感器。磁性液体传感器应用领域很广，既可以应用到民用上，也可以应用到军工上，有其他传感器所代替不了的功能，正因如此，国外比较早地意识到开发磁性液体传感器的意义，并且已开始了研究和生产，并将该种传感器应用到航空、航天、宇航站等尖端军事领域。美国、法国、德国、俄罗斯、日本和罗马尼亚等国家已开始利用磁性液体制作各种传感器。磁性液体水平传感器对控制机器人工作状态，使太阳能栅板保持朝向太阳，使抛物天线持续朝向通信系统中的人造卫星等方面有着重要的应用。目前国外已研制出单轴、双轴、三轴等类型的磁性液体水平传感器，而我国有关磁性液体传感器的研究尚处于实验和探索阶段。

5.4.3 Android 系统中的磁场传感器

在 Android 系统中，磁场传感器 TYPE_MAGNETIC_FIELD，单位是 μT（微特斯拉），能够测量设备周围 3 个物理轴（x，y，z）的磁场。在 Android 设备中，磁场传感器主要用于感应周围的磁感应强度，在注册监听器后主要用于捕获如下 3 个参数：

- values[0]
- values[l]
- values[2]

上述 3 个参数分别代表磁感应强度在空间坐标系中 3 个方向轴上的分量。所有数据的单位为 μT，即微特斯拉。

在 Android 系统中，磁场传感器主要包含了如下公共方法。

- int getFifoMaxEventCount()：返回该传感器可以处理事件的最大值。如果该值为0，表示当前模式不支持此传感器。
- int getFifoReservedEventCount()：保留传感器在批处理模式中FIFO的事件数，给出了一个在保证可以分批事件的最小值。
- float getMaximumRange()：传感器单元的最大范围。
- int getMinDelay()：最小延迟。
- String getName()：获取传感器的名称。
- float getPower()：获取传感器电量。
- float getResolution()：获得传感器的分辨率。
- int getType()：获取传感器的类型。
- String getVendor()：获取传感器的供应商字符串。
- int getVersion()：获取该传感器模块版本。
- String toString()：返回一个对当前传感器的字符串描述。

5.5 加速度传感器详解

知识点讲解：光盘:视频\知识点\第 5 章\加速度传感器详解.avi

在现实应用中，加速度传感器可以帮助机器人了解它现在身处的环境，能够分辨出是在登山还是

在下山、是否摔倒等。一个好的程序员能够使用加速度传感器辨出上述情形，加速度传感器甚至可以用来分析发动机的振动。本节将简要讲解 Android 系统中加速度传感器的基础性知识。

5.5.1 加速度传感器的分类

在实际应用过程中，可以将加速度传感器分为如下 4 类。

（1）压电式

压电式加速度传感器又称压电加速度计。它也属于惯性式传感器。压电式加速度传感器的原理是利用压电陶瓷或石英晶体的压电效应，在加速度计受振时，质量块加在压电元件上的力也随之变化。当被测振动频率远低于加速度计的固有频率时，则力的变化与被测加速度成正比。

（2）压阻式

基于世界领先的 MEMS 硅微加工技术，压阻式加速度传感器具有体积小、低功耗等特点，易于集成在各种模拟和数字电路中，广泛应用于汽车碰撞实验、测试仪器、设备振动监测等领域。加速度传感器网为客户提供压阻式加速度传感器/压阻加速度计各品牌的型号、参数、原理、价格、接线图等信息。

（3）电容式

电容式加速度传感器是基于电容原理的极距变化型的电容传感器。电容式加速度传感器/电容式加速度计是对比较通用的加速度传感器，在某些领域无可替代，如安全气囊和手机移动设备等。电容式加速度传感器/电容式加速度计采用了微机电系统（MEMS）工艺，在大量生产时变得经济，从而保证了较低的成本。

（4）伺服式

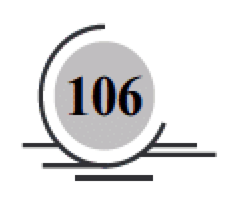
伺服式加速度传感器是一种闭环测试系统，具有动态性能好、动态范围大和线性度好等特点。其工作原理是，传感器的振动系统由 m-k 系统组成，与一般加速度计相同，但质量 m 上还接着一个电磁线圈，当基座上有加速度输入时，质量块偏离平衡位置，该位移大小由位移传感器检测出来，经伺服放大器放大后转换为电流输出，该电流流过电磁线圈，在永久磁铁的磁场中产生电磁恢复力，力图使质量块保持在仪表壳体中原来的平衡位置上，所以伺服加速度传感器在闭环状态下工作。由于有反馈作用，增强了抗干扰的能力，提高测量精度，扩大了测量范围，伺服加速度测量技术广泛地应用于惯性导航和惯性制导系统中，在高精度的振动测量和标定中也有应用。

5.5.2 加速度传感器的主要应用领域

在计算机领域中，加速度传感器可以测量牵引力产生的加速度。例如在 IBM Thinkpad 笔记本电脑中就内置了加速度传感器，能够动态地监测出笔记本在使用中的振动。根据这些振动数据，系统会智能地选择关闭硬盘还是让其继续运行，这样可以最大程度地保护由于振动而引发的硬件损坏问题。所以，加速度传感器主要应用在手柄振动/摇晃、仪器仪表、汽车制动启动、地震、报警系统、玩具、结构物、环境监视、工程测振、地质勘探、铁路、桥梁、大坝的振动测试与分析，还有鼠标，高层建筑结构动态特性和安全保卫振动侦察方面。在接下来的内容中，将详细讲解加速度传感器的主要应用领域。

（1）汽车安全

加速度传感器主要用于汽车安全气囊、防抱死系统、牵引控制系统等安全性能方面。在汽车安全



应用中，加速度计的快速反应非常重要。安全气囊应在什么时候弹出要迅速确定，所以加速度计必须在瞬间作出反应。通过采用可迅速达到稳定状态而不是振动不止的传感器设计可以缩短器件的反应时间。其中，压阻式加速度传感器由于在汽车工业中的广泛应用而发展最快。

（2）游戏控制

加速度传感器可以检测上下左右的倾角变化，因此通过前后倾斜手持设备来实现对游戏中物体的前后左右的方向控制，就变得很简单了。

（3）图像自动翻转

用加速度传感器检测手持设备的旋转动作及方向，实现所要显示图像的转正。

（4）电子指南针倾斜校正

磁传感器是通过测量磁通量的大小来确定方向的。当磁传感器发生倾斜时，通过磁传感器的地磁通量将发生变化，从而使方向指向产生误差。因此，如果不带倾斜校正的电子指南针，需要用户水平放置。而利用加速度传感器可以测量倾角的这一原理，可以对电子指南针的倾斜进行补偿。

（5）GPS 导航系统死角的补偿

GPS 系统是通过接收三颗呈 120°分布的卫星信号来最终确定物体的方位的。在一些特殊的场合和地貌，如隧道、高楼林立、丛林地带，GPS 信号会变弱甚至完全消失，这也就是所谓的死角。而通过加装加速度传感器及以前我们所通用的惯性导航，便可以进行系统死区的测量。对加速度传感器进行一次积分，就变成了单位时间里的速度变化量，从而测出在死区内物体的移动。

（6）计步器功能

加速度传感器可以检测交流信号以及物体的振动，人在走动的时候会产生一定规律性的振动，而加速度传感器可以检测振动的过零点，从而计算出人所走的步数或跑步所跑的步数，从而计算出人所移动的位移，并且利用一定的公式可以计算出卡路里的消耗。

（7）防手抖功能

用加速度传感器检测手持设备的振动/晃动幅度，当振动/晃动幅度过大时锁住照相快门，使所拍摄的图像永远是清晰的。

（8）闪信功能

通过挥动手持设备实现在空中显示文字，用户可以自己编写显示的文字。这个闪信功能是利用人们的视觉残留现象，用加速度传感器检测挥动的周期，实现所显示文字的准确定位。

（9）硬盘保护

利用加速度传感器检测自由落体状态，从而对迷你硬盘实施必要的保护。大家知道，硬盘在读取数据时，磁头与碟片之间的间距很小，因此，外界的轻微振动就会对硬盘产生很坏的后果，使数据丢失，而利用加速度传感器可以检测自由落体状态。当检测到自由落体状态时，让磁头复位，以减少硬盘的受损程度。

（10）设备或终端姿态检测

加速度传感器和陀螺仪通常称为惯性传感器，常用于各种设备或终端中实现姿态检测、运动检测等，很适合玩体感游戏的人群。加速度传感器利用重力加速度，可以用于检测设备的倾斜角度，但是它会受到运动加速度的影响，使倾角测量不够准确，所以通常需利用陀螺仪和磁传感器补偿。同时磁传感器测量方位角时，也是利用地磁场，当系统中电流变化或周围有导磁材料时，以及当设备倾斜时，测量出的方位角也不准确，这时需要用加速度传感器（倾角传感器）和陀螺仪进行补偿。

（11）智能产品

加速度传感器在微信功能中的创新功能突破了电子产品的千篇一律，这个功能的实现来源于传感器的方向、加速表、光线、磁场、临近性、温度等参数的特性。这个原理是手机里面集成的加速度传感器，它能够分别测量 X、Y、Z 三个方面的加速度值，X 方向值的大小代表手机水平移动，Y 方向值的大小代表手机垂直移动，Z 方向值的大小代表手机的空间垂直方向，天空的方向为正，地球的方向为负，然后把相关的加速度值传输给操作系统，通过判断其大小变化，就能知道同时玩微信的朋友。

5.5.3 线性加速度传感器的原理

线性加速度传感器是加速度传感器的一种，分开单独讲的原因是为了和陀螺仪传感器区分开。陀螺仪是测角速度的，加速度是测线性加速度的。其中前者利用了惯性原理，而后者利用了力平衡原理。线性加速度传感器利用了惯性原理：

```
A（加速度）=F（惯性力）/M（质量）
```

我们只需要测量 F 即可。怎么测量 F？是要电磁力去平衡这个力就可以得到 F 对应于电流的关系，只需要用实验去标定这个比例系数就行了。多数加速度传感器是根据压电效应的原理来工作的。所谓的压电效应就是“对于不存在对称中心的异极晶体加在晶体上的外力除了使晶体发生形变以外，还将改变晶体的极化状态，在晶体内部建立电场，这种由于机械力作用使介质发生极化的现象称为正压电效应”。

一般加速度传感器就是利用了其内部的由于加速度造成的晶体变形这个特性。由于这个变形会产生电压，只要计算出产生电压和所施加的加速度之间的关系，就可以将加速度转化成电压输出。当然，还有很多其他方法来制作加速度传感器，比如压阻技术、电容效应、热气泡效应、光效应，但是其最基本的原理都是由于加速度产生某个介质产生变形，通过测量其变形量并用相关电路转化成电压输出。每种技术都有各自的机会和问题。

压阻式加速度传感器由于在汽车工业中的广泛应用而发展最快。由于安全性越来越成为汽车制造商的卖点，这种附加系统也越来越多。根据有关调查，预计其市值将按年平均 4.1%速度增长，这其中欧洲市场的速度最快，因为欧洲是许多安全气囊和汽车生产企业的所在地。

压电技术主要在工业上用来防止机器故障，使用这种传感器可以检测机器潜在的故障以达到自保护，及避免对工人产生意外伤害，这种传感器具有用户尤其是质量行业的用户所追求的可重复性、稳定性和自生性。但是在许多新的应用领域，很多用户尚无使用这类传感器的意识，销售商冒险进入这种尚待开发的市场会麻烦多多，因为终端用户对由于使用这种传感器而带来的问题和解决方法都认识不多。如果这些问题能够得到解决，将会促进压电传感器得到更快的发展。

使用加速度传感器有时会碰到低频场合测量时输出信号出现失真的情况，用多种测量判断方法一时找不出故障出现的原因，经过分析总结，导致测量结果失真的因素主要是：系统低频响应差、系统低频信噪比差、外界环境对测量信号的影响。所以，只要出现加速度传感器低频测量信号失真情况，对比以上 3 点看看是哪个因素造成的，有针对性地去解决。

在 Android 系统中，线性加速度传感器的类型是 TYPE_LINEAR_ACCELERATION，单位是 m/s^2，能够获取加速度传感器去除重力的影响得到的数据。

5.5.4　Android 系统中的加速度传感器

在 Android 系统中，加速度传感器是 TYPE_ACCELEROMETER，单位是 m/s^2，能够测量应用于设备 X、Y、Z 轴上的加速度，又叫做 G-sensor。在开发过程中，通过 Android 的加速度传感器可以取得 X、Y、Z 三个轴的加速度。在 Android 系统中，在类 SensorManager 中定义了很多星体的重力加速度值，如表 5-1 所示。

表 5-1　类 SensorManager 被定义的各新星体的重力加速度值

常　量　名	说　　明	实 际 的 值
GRAVITY_DEATH_STAR_1	死亡星	3.5303614E-7
GRAVITY_EARTH	地球	9.80665
GRAVITY_JUPITER	木星	23.12
GRAVITY_MARS	火星	3.71
GRAVITY_MERCURY	水星	3.7
GRAVITY_MOON	月亮	1.6
GRAVITY_NEPTUNE	海王星	12.0
GRAVITY_PLUTO	冥王星	0.6
GRAVITY_SATURN	土星	8.96
GRAVITY_SUN	太阳	275.0
GRAVITY_THE_ISLAND	岛屿星	4.815162
GRAVITY_URANUS	天王星	8.69
GRAVITY_VENUS	金星	8.87

通常来说，从加速度传感器获取的值，拿手机等智能设备的人的手震动或放在摇晃的场所的时候，受震动影响设备的值增幅变化是存在的。手的摇动、轻微震动的影响是属于长波形式，去掉这种长波干扰的影响，可以取得高精度的值。去掉这种长波的过滤机能叫 Low-Pass Filter。Low-Pass Filter 机制有如下 3 种封装方法。

- ❑ 从抽样数据中取得中间的值的方法。
- ❑ 最近取得的加速度的值每个很少变化的方法。
- ❑ 从抽样数据中取得中间的值的方法。

在 Android 应用中，有时需要获取瞬间加速度值，例如类似计步器、作用力测定的应用开发的时候，如果想检测出加速度急剧的变化。此时的处理和 Low-Pass Filter 处理相反，需要去掉短周波的影响，这样可以取得数据。像这种去掉短周波的影响的过滤器叫做 High-pass Filter。

5.6　方向传感器详解

知识点讲解：光盘:视频\知识点\第 5 章\方向传感器详解.avi

在 Android 设备中，经常需要检测设备的方向，例如设备的朝向和移动方向。在 Android 系统中，通常使用重力传感器、加速度传感器、磁场传感器和旋转矢量传感器来检测设备的方向。本节内容将详细讲解在 Android 设备中使用方向传感器检测设备方向的基本知识，为读者学习本书后面的知识打下基础。

5.6.1 方向传感器基础

在现实世界中，方向传感器通过对力敏感的传感器，感受手机等设备在变换姿势时的重心变化，使手机等设备光标变化位置从而实现选择的功能。方向传感器运用了欧拉角的知识，欧拉角的基本思想是将角位移分解为绕 3 个互相垂直轴的 3 个旋转组成的序列。其实，任意 3 个轴和任意顺序都可以，但最有意义的是使用笛卡儿坐标系并按一定的顺序所组成的旋转序列。

在学习欧拉角知识之前先介绍几种不同概念的坐标系，以便于读者理解欧拉角知识。

（1）世界坐标系

世界坐标系是一个特殊的坐标系，建立了描述其他坐标系所需要的参考框架。能够用世界坐标系描述其他坐标系的位置，而不能用更大的、外部的坐标系来描述世界坐标系。例如，“向西”“向东”等词汇就是世界坐标系中的描述词汇。

（2）物体坐标系

物体坐标系是和特定物体相关联的坐标系，每个物体都有它们独立的坐标系。当物体移动或改变方向时，和该物体相关联的坐标系将随之移动或改变方向。例如，“向左”“向右”等词汇就是物体坐标系中的描述词汇。

（3）摄像机坐标系

摄像机坐标系是和观察者密切相关的坐标系。在摄像机坐标系中，摄像机在原点，x 轴向右，z 轴向前（朝向屏幕内或摄像机方向），y 轴向上（不是世界的上方而是摄像机本身的上方）。

（4）惯性坐标系

惯性坐标系是为了简化世界坐标系到物体坐标系的转换而引入的一种新的坐标系。惯性坐标系的原点和物体坐标系的原点重合，但惯性坐标系的轴平行于世界坐标系的轴。

在欧拉角中，表示一个物体的方位用“Yaw-Pitch-Roll”约定。在这个系统中，一个方位被定义为一个 Yaw 角、一个 Pitch 角和一个 Roll 角。欧拉角的基本思想是让物体开始于“标准”方位，目的是使物体坐标轴和惯性坐标轴对齐。在标准方位上，让物体做 Yaw、Pitch 和 Roll 旋转，最后物体到达我们想要描述的方位。

（5）Yaw 轴

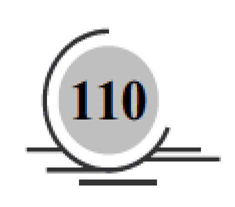
Yaw 轴是 3 个方向轴中唯一不变的轴，其方向总是竖直向上，和世界坐标系中的 z 轴是等同的，也就是重力加速度 g 的反方向。

（6）Pitch 轴

Pitch 轴方向依赖于手机沿 Yaw 轴的转动情况，即当手机沿 Yaw 转过一定的角度后，Pitch 轴也相应围绕 Yaw 轴转动相同的角度。Pitch 轴的位置依赖于手机沿 Yaw 轴转过的角度，好比 Yaw 轴和 Pitch 轴是两根焊死在一起成 90°。

5.6.2 Android 中的方向传感器

在 Android 系统中，方向传感器的类型是 TYPE_ORIENTATION，用于测量设备围绕 3 个物理轴（X，Y，Z）的旋转角度，在新版本中已经使用 SensorManager.getOrientation()替代。Android 系统中的方向传感器在生活中的典型应用例子是指南针，接下来先来简单介绍一下传感器中 3 个参数 X、Y、Z 的含义，如图 5-22 所示。



如图 5-22 所示，浅色部分表示一个手机，带有小圈那一头是手机头部，各个部分的具体说明如下。

- 传感器中的X：如图5-22所示，规定X正半轴为北，手机头部指向OF方向，此时X的值为0。如果手机头部指向OG方向，此时X值为90；指向OH方向，X值为180；指向OE方向，X值为270。
- 传感器中的Y：现在将手机沿着BC轴慢慢向上抬起，即手机头部不动，尾部慢慢向上翘起来，直到AD跑到BC右边并落在XOY平面上，Y的值将在0～180之间变动，如果手机沿着AD轴慢慢向上抬起，即手机尾部不动，直到BC跑到AD左边并且落在XOY平面上，Y的值将在0～-180之间变动，这就是方向传感器中Y的含义。

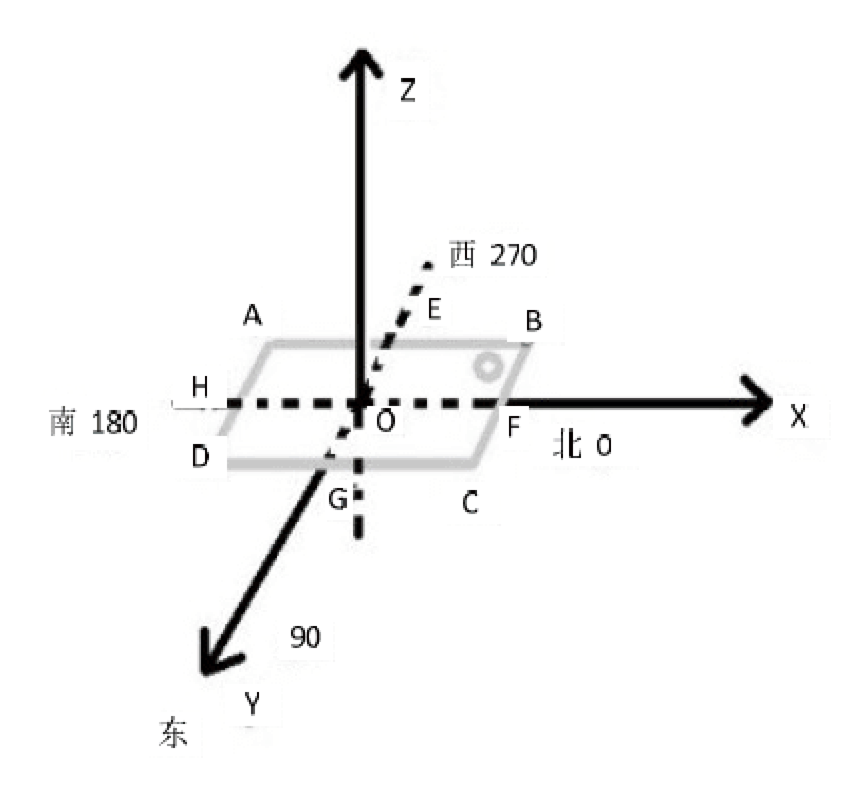


图 5-22　参数 X、Y、Z

- 传感器中的Z：现在将手机沿着AB轴慢慢向上抬起，即手机左边框不动，右边框慢慢向上翘起来，直到CD跑到AB右边并落在XOY平面上，Z的值将从0~180之间变动，如果手机沿着CD轴慢慢向上抬起，即手机右边框不动，直到AB跑到CD左边并且落在XOY平面上，Z的值将在0～-180之间变动，这就是方向传感器中Z的含义。

5.7　陀螺仪传感器详解

知识点讲解：光盘:视频\知识点\第 5 章\陀螺仪传感器详解.avi

陀螺仪传感器是一个基于自由空间移动和手势的定位和控制系统。例如我们可以在假想的平面上移动鼠标，屏幕上的光标就会随之跟着移动，并且可以绕着链接画圈和点击按键。又比如当我们正在演讲或离开桌子时，这些操作都能够很方便地实现。陀螺仪传感器已经被广泛运用于手机、平板等移动便携设备上，将来的设备也会陆续使用陀螺仪传感器。本节将详细讲解在 Android 设备中使用陀螺仪传感器的基本知识，为读者学习本书后面的知识打下基础。

5.7.1　陀螺仪传感器基础

陀螺仪的原理是，当一个旋转物体的旋转轴所指的方向在不受外力影响时是不会改变的。根据这个道理，可以用陀螺仪来保持方向。然后用多种方法读取轴所指示的方向，并自动将数据信号传给控制系统。在现实生活中，骑自行车便是利用了这个原理。轮子转得越快越不容易倒，因为车轴有一种保持水平的力量。现代陀螺仪可以精确地确定运动物体的方位，在现代航空、航海、航天和国防工业中广泛使用的一种惯性导航仪器。传统的惯性陀螺仪主要部分有机械式的陀螺仪，而机械式的陀螺仪对工艺结构的要求很高。20 世纪 70 年代提出了现代光纤陀螺仪的基本设想，到 80 年代以后，光纤陀螺仪就得到了非常迅速的发展，激光谐振陀螺仪也有了很大的发展。光纤陀螺仪具有结构紧凑、灵敏度高、工作可靠的特点，在很多的领域已经完全取代了机械式的传统的陀螺仪，成为现代导航仪器中的关键部件。

根据框架的数目和支承的形式以及附件的性质进行划分，陀螺仪传感器的主要类型有以下两种。

（1）二自由度陀螺仪：只有一个框架，使转子自转轴具有一个转动自由度。根据二自由度陀螺仪

中所使用的反作用力矩的性质，可以把这种陀螺仪分为如下 3 种类型。

- ❑ 积分陀螺仪（它使用的反作用力矩是阻尼力矩）。
- ❑ 速率陀螺仪（它使用的反作用力矩是弹性力矩）。
- ❑ 无约束陀螺仪（它仅有惯性反作用力矩）。

另外，除了机、电框架式陀螺仪外还出现了某些新型陀螺仪，例如静电式自由转子陀螺仪、挠性陀螺仪和激光陀螺仪等。

（2）三自由度陀螺仪：具有内、外两个框架，使转子自转轴具有两个转动自由度。在没有任何力矩装置时，它就是一个自由陀螺仪。

在当前技术水平条件下，陀螺仪传感器主要被用于如下两个领域。

（1）国防工业

陀螺仪传感器原本是运用到直升机模型上的，而今已经被广泛运用于手机类移动便携设备上，不仅仅如此，现代陀螺仪是一种能够精确地确定运动物体的方位的仪器，所以陀螺仪传感器是现代航空、航海、航天和国防工业应用中必不可少的控制装置。陀螺仪传感器是法国的物理学家莱昂·傅科在研究地球自转时命名的，到如今一直是航空和航海上航行姿态及速率等最方便实用的参考仪表。

（2）开门报警器

陀螺仪传感器可以测量开门的角度，当门被打开一个角度后会发出报警声，或者结合 GPRS 模块发送短信以提醒门被打开了。另外，陀螺仪传感器集成了加速度传感器的功能，当门被打开的瞬间，将产生一定的加速度值，陀螺仪传感器将会测量到这个加速度值，达到预设的门槛值后，将发出报警声，或者结合 GPRS 模块发送短信以提醒门被打开了。报警器内还可以集成雷达感应测量功能，如要有人进入房间内移动时就会被雷达测量到。这种双重保险提醒防盗，可靠性高，误报率低，非常适合重要场合的防盗报警。

5.7.2 Android 中的陀螺仪传感器

在 Android 系统中，陀螺仪传感器的类型是 TYPE_GYROSCOPE，单位是 rad/s，能够测量设备 X、Y、Z 三轴的角加速度数据。Android 中的陀螺仪传感器又名为 Gyro-sensor 角速度器，利用内部震动机械结构侦测物体转动所产生的角速度，进而计算出物体移动的角度。侦测水平改变的状态，但无法计算移动的激烈程度。在接下来的内容中，将详细讲解 Android 中的陀螺仪传感器的基本知识。

（1）陀螺仪传感器和加速度传感器的对比

在 Android 的传感器系统中，陀螺仪传感器和加速度传感器非常类似，两者的区别如下。

- ❑ 加速度传感器：用于测量加速度，借助一个三轴加速度计可以测得一个固定平台相对地球表面的运动方向，但是一旦平台运动起来，情况就会变得复杂得多。如果平台做自由落体，加速度计测得的加速度值为0。如果平台朝某个方向做加速度运动，各个轴向加速度值会含有重力产生的加速度值，使得无法获得真正的加速度值。例如，安装在60°横滚角飞机上的三轴加速度计会测得2g的垂直加速度值，而事实上飞机相对地区表面是60°的倾角。因此，单独使用加速度计无法使飞机保持一个固定的航向。
- ❑ 陀螺仪传感器：用于测量机体围绕某个轴向的旋转角速率值。当使用陀螺仪测量飞机机体轴向的旋转角速率时，如果飞机在旋转，测得的值为非零值，飞机不旋转时，测量的值为0。因此，在60°横滚角的飞机上的陀螺仪测得的横滚角速率值为0，同样在飞机做水平直线飞行时的角速率值为0。

可以通过角速率值的时间积分来估计当前的横滚角度，前提是没有误差的累积。陀螺仪测量的值会随时间漂移，经过几分钟甚至几秒钟定会累积出额外的误差来，而最终会导致对飞机当前相对水平面横滚角度完全错误的认知。因此，单独使用陀螺仪也无法保持飞机的特定航向。

综上所述，加速度传感器在较长时间的测量值（确定飞机航向）是正确的，而在较短时间内由于信号噪声的存在而有误差。陀螺仪传感器在较短时间内则比较准确，而较长时间则会由于漂移而存有误差。因此，需要两者（相互调整）来确保航向的正确。

（2）外设项目开发过程中的陀螺仪传感器

在外设项目开发过程中，三自由度陀螺仪是一个可以识别设备，能够相对于地面绕 X、Y、Z 轴转动角度的感应器（自己的理解，不够严谨）。无论是可移动设备，还是智能手机、平板电脑，通过使用陀螺仪传感器可以实现很多好玩的应用，例如说指南针。

在实际开发过程中，可以用一个磁场感应器（Magnetic Sensor）来实现陀螺仪。磁场感应器是用来测量磁场感应强度的。一个 3 轴的磁 sensor IC 可以得到当前环境下 X、Y 和 Z 方向上的磁场感应强度，对于 Android 中间层来说就是读取该感应器测量到的这 3 个值。当需要时，上报给上层应用程序。磁感应强度的单位是 T（特斯拉）或者是 Gs（高斯），1T 等于 10000Gs。

在了解陀螺仪之前，需要先了解 Android 系统定义坐标系的方法，如下所示的文件中进行了定义。

```
/hardware/libhardware/include/hardware/sensors.h
```

在上述文件 sensors.h 中，有一个如图 5-23 所示的效果图。

图 5-23 中表示设备的正上方是 Y 轴方向，右边是 X 轴方向，垂直设备屏幕平面向前的是 Z 轴方向，这个很重要。因为应用程序就是根据这样的定义来写的，所以我们报给应用的数据要跟这个定义符合。还需要清楚磁 Sensor 芯片贴在板上的坐标系。我们从芯片读出数据后要把芯片的坐标系转换为设备的实际坐标系。除非芯片贴在板上刚好跟设备的 X、Y、Z 轴方向一致。

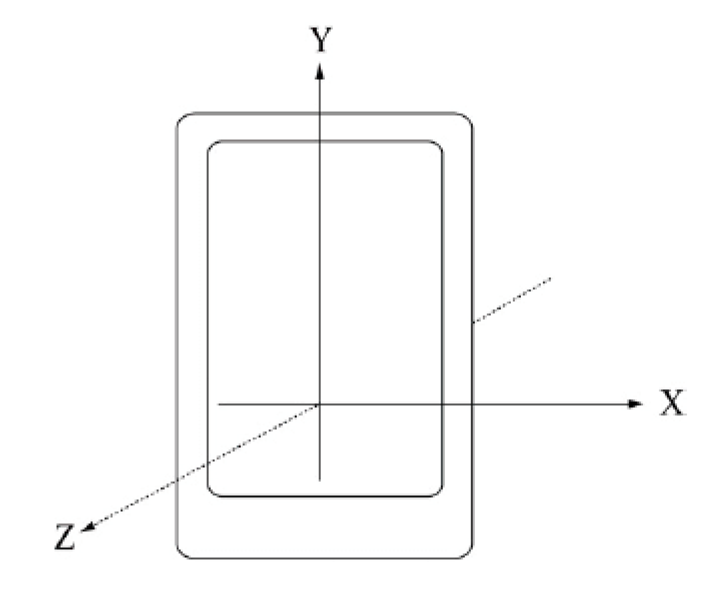


图 5-23　Android 系统定义的坐标系

陀螺仪的实现是根据磁场感应强度的 3 个值计算出另外 3 个值。当需要时可以计算出这 3 个值上报给应用程序，这样就实现了陀螺仪的功能。在接下来的内容中，将详细讲解这 3 个值的具体含义和计算方法。

（1）Azimuth 方位角

Azimuth 方位角，即绕 Z 轴转动的角度，0° =正北。假设 Y 轴指向地磁正北方，直升机正前方的方向如图 5-24 所示，90° =正东时如图 5-25 所示。

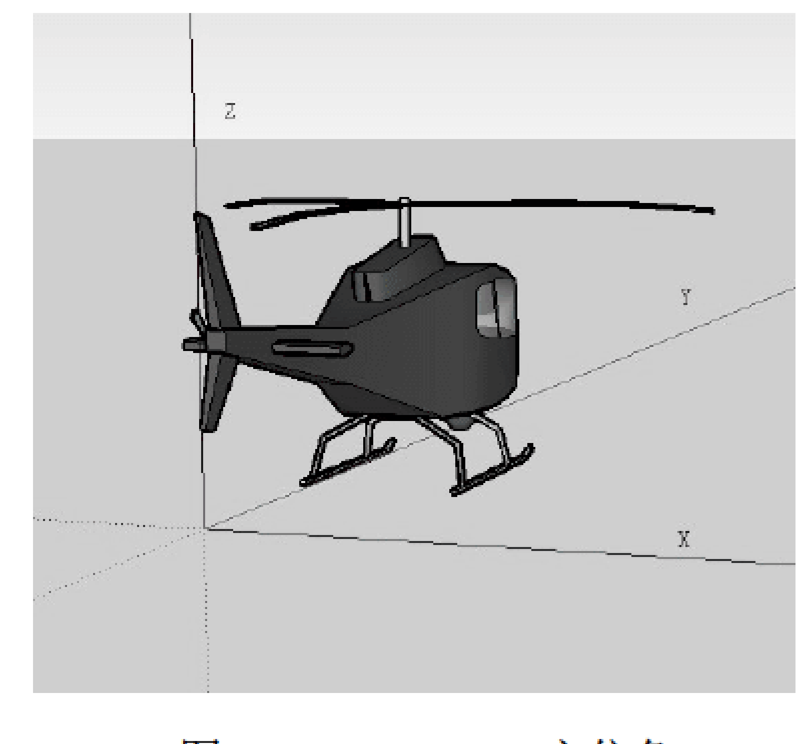


图 5-24　Azimuth 方位角

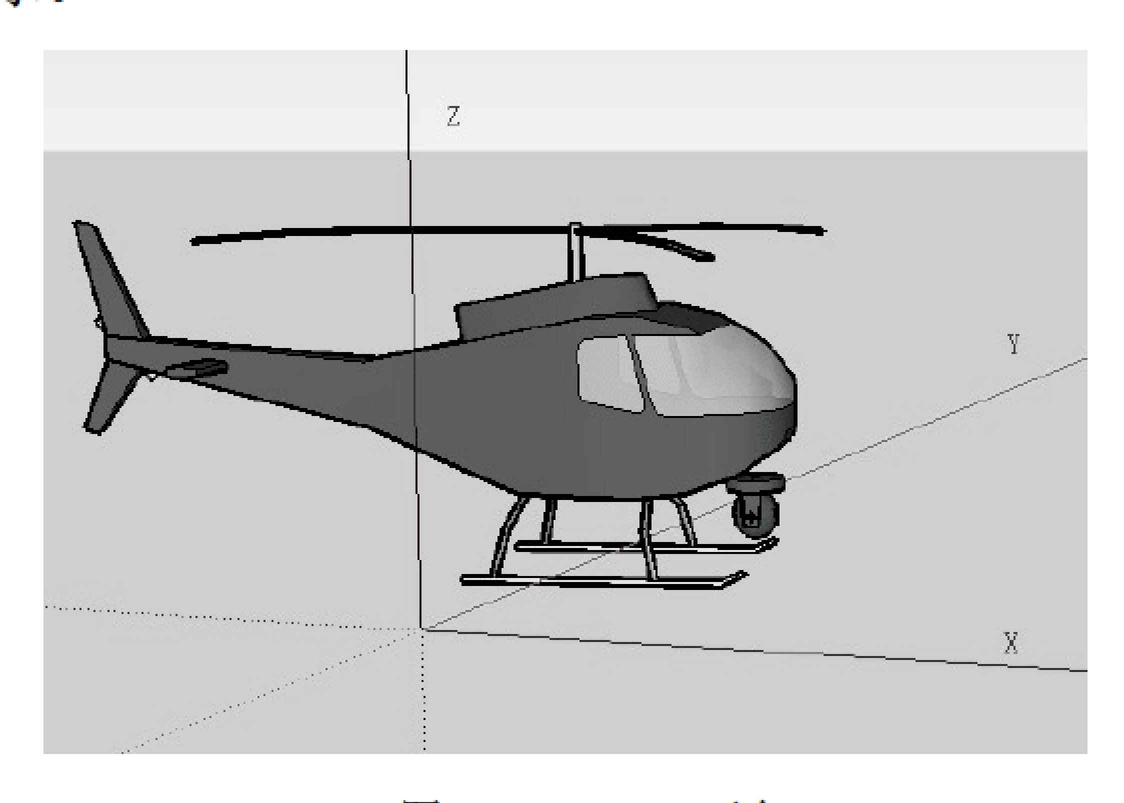


图 5-25　90° =正东

180° =正南时如图 5-26 所示，270° =正西时如图 5-27 所示。

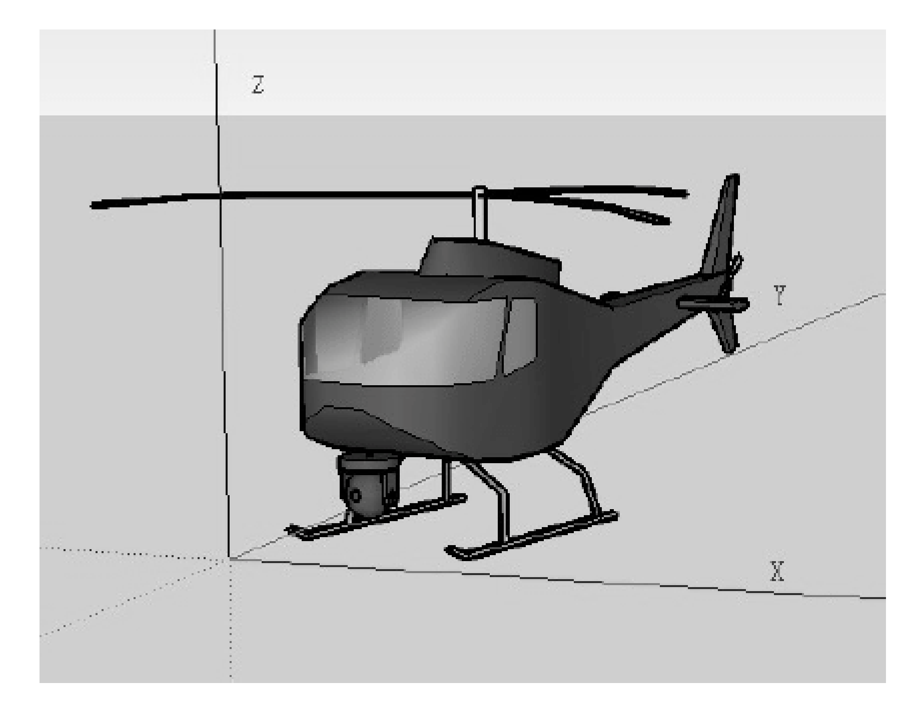



图 5-26　180° =正南

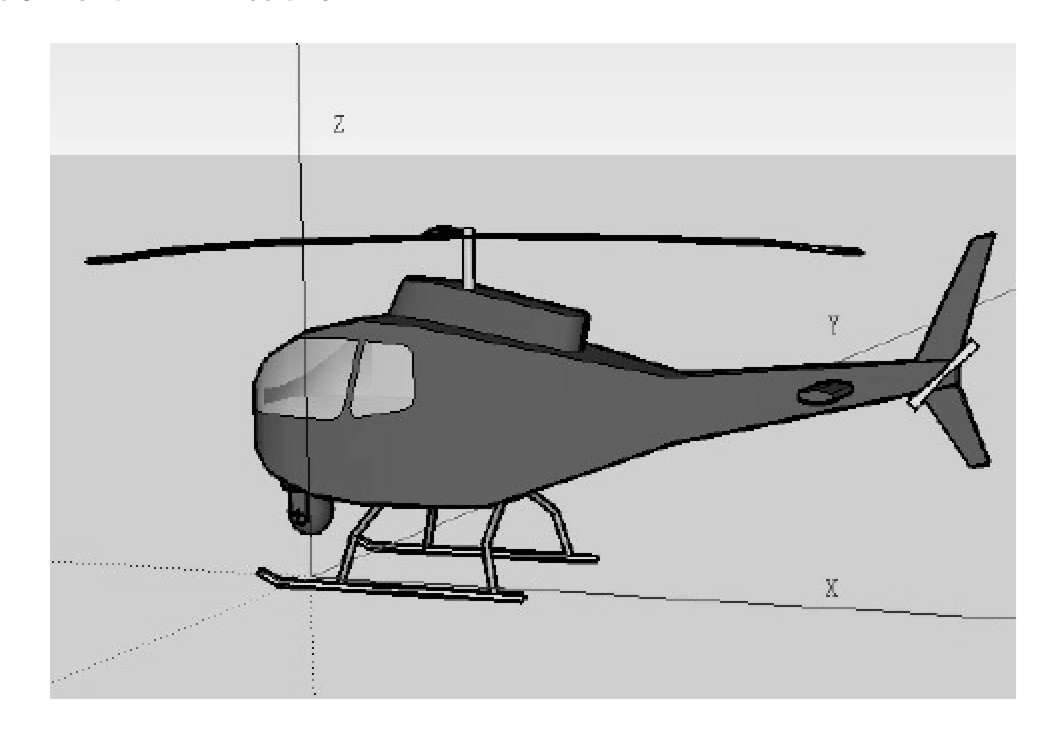


图 5-27　270° =正西

在这种情况下，通过计算 X 和 Y 方向的磁感应强度的反正切的方式就可以得到方位角。要想实现指南针，只需要用这个值即可（不考虑设备非水平的情况）。

（2）Pitch 仰俯

绕 X 轴转动的角度（-180<=pitch<=180），如果设备水平放置，前方向下俯就是正，如图 5-28 所示。前方向上仰就是负值，如图 5-29 所示。

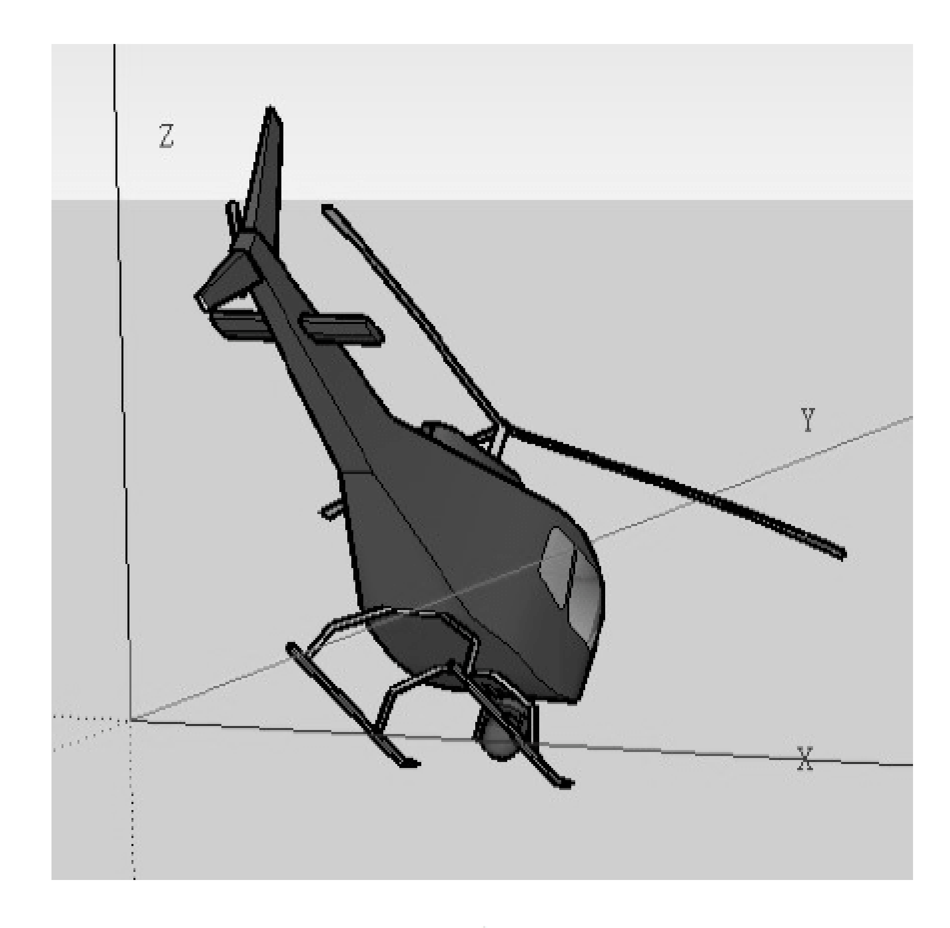


图 5-28　前方向下俯就是正

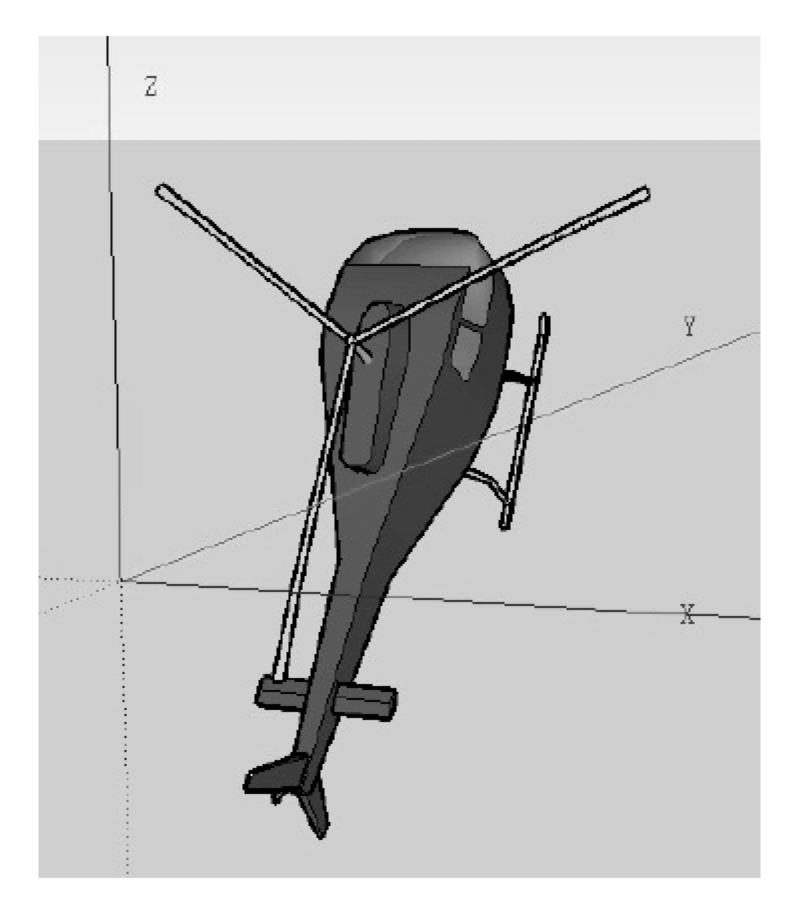


图 5-29　前方向上仰就是负值

在这种情况下，计算磁 Sensor 的 Y 和 Z 反正切就可以得到此角度值，如图 5-30 所示。

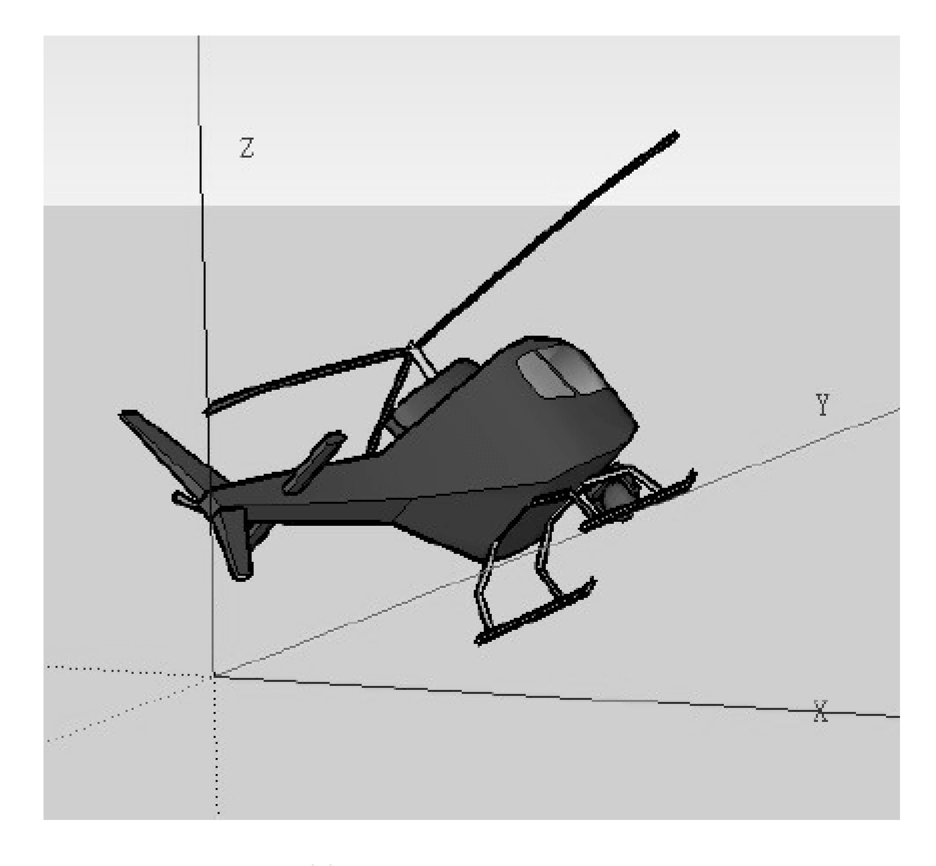


图 5-30　计算磁 Sensor 的 Y 和 Z 反正切

5.8　距离传感器详解

知识点讲解：光盘:视频\知识点\第 5 章\距离传感器详解.avi

在 Android 设备应用程序开发过程中，经常需要检测设备的运动数据，例如设备的运动速率和运动距离等。这些数据对于健身类设备来说都是十分重要的，例如健身手表可以及时测试晨练的运动距离和速率。在 Android 系统中，通常使用加速度传感器、线性加速度传感器和距离传感器来检测设备的运动数据。本节将详细讲解在 Android 设备中检测运动数据的基本知识。

5.8.1　距离传感器介绍

在 Android 系统中，需要使用加速度传感器、线性加速度传感器和距离传感器来检测设备的运动数据。在当前的技术条件下，距离传感器是指利用“飞行时间法”（flying time）的原理来实现测量距离，以实现检测物体距离的一种传感器。“飞行时间法”是通过发射特别短的光脉冲，并测量此光脉冲从发射到被物体反射回来的时间，通过测时间间隔来计算与物体之间的距离。

在现实世界中，距离传感器在智能手机中的应用比较常见。一般触屏智能手机在默认设置下，都会有一个延时锁屏的设置，就是在一段时间内，手机检测不到任何操作，就会进入锁屏状态。这样是有一定好处的。手机作为移动终端的一种，追求低功耗是设计的目标之一。延时锁屏既可以避免不必要的能量消耗，又能保证不丢失重要信息。另外，在使用触屏手机设备时，当接电话的时候距离传感器会起作用，当脸靠近屏幕时屏幕灯会熄灭，并自动锁屏，这样可以防止脸误操作。当脸离开屏幕时屏幕灯会自动开启，并且自动解锁。

除了被广泛应用于手机设备之外，距离传感器还被用于野外环境（山体情况、峡谷深度等）、飞机高度检测、矿井深度、物料高度测量等领域，并且在野外应用领域中，主要用于检测山体情况和峡谷深度等。而对飞机高度测量功能是通过检测飞机在起飞和降落时距离地面的高度，并将结果实时显示在控制面板上；也可以使用距离传感器测量物料各点高度，用于计算物料的体积。在显示应用中，用于飞机高度和物料高度的距离传感器有 LDM301 系列，用于野外应用的距离传感器有 LDM4x 系列。

在当前的可设备应用中，距离传感器被应用于智能皮带中。在皮带扣里嵌入了距离传感器，当把皮带调整至合适宽度、卡好皮带扣后，如果皮带在 10 秒钟内没有重新解开，传感器就会自动生成本次的腰围数据。皮带与皮带扣连接处的其中一枚铆钉将被数据传输装置所替代。当你将智能手机放在铆钉处保持两秒钟静止，手机里的自我健康管理 App 会被自动激活，并获取本次腰围数据。

5.8.2　Android 系统中的距离传感器

在 Android 系统中，距离传感器也被称为 P-Sensor，值是 TYPE_PROXIMITY，单位是 cm，能够测量某个对象到屏幕的距离。可以在打电话时判断人耳到电话屏幕的距离，以关闭屏幕而达到省电功能。

P-Sensor 主要用于在通话过程中防止用户误操作屏幕，接下来以通话过程为例来讲解电话程序对 P-Sensor 的操作流程。

（1）在启动电话程序的时候，在“.java”文件中新建了一个 P-Sensor 的 WakeLock 对象，其代码

如下所示。

```
mProximityWakeLock =    pm.newWakeLock(
PowerManager.PROXIMITY_SCREEN_OFF_WAKE_LOCK, LOG_TAG
);
```

对象 wackLock 的功能是请求控制屏幕的点亮或熄灭。

（2）在电话状态发生改变时，例如接通了电话，调用“.java”文件中的方法根据当前电话的状态来决定是否打开 P-Sensor。如果在通话过程中，电话是 OFF-HOOK 状态时打开 P-Sensor，其演示代码如下。

```
if (!mProximityWakeLock.isHeld()) {
                        if (DBG) Log.d(LOG_TAG, "updateProximitySensorMode: acquiring...");
                        mProximityWakeLock.acquire();
                    }
```

在上述代码中，mProximityWakeLock.acquire()会调用到另外的方法打开 P-Sensor，这个另外的方法会判断当前手机有没有 P-Sensor。如果有，就会向 SensorManager 注册一个 P-Sensor 监听器。这样当 P-Sensor 检测到手机和人体距离发生改变时，就会调用服务监听器进行处理。同样，当电话挂断时，电话模块会去调用方法取消 P-Sensor 监听器。

在 Android 系统中，PowerManagerService 中 P-Sensor 监听器会进行实时监听工作，当 P-Sensor 检测到距离有变化时就会进行监听。具体监听过程的代码如下所示。

```
SensorEventListener mProximityListener = new SensorEventListener() {
        public void onSensorChanged(SensorEvent event) {
            long milliseconds = SystemClock.elapsedRealtime();
            synchronized (mLocks) {
                float distance = event.values[0];   //检测到手机和人体的距离
                long timeSinceLastEvent = milliseconds - mLastProximityEventTime;   //这次检测和上次检测
的时间差
                mLastProximityEventTime = milliseconds;   //更新上一次检测的时间
                mHandler.removeCallbacks(mProximityTask);
                boolean proximityTaskQueued = false;

                // compare against getMaximumRange to support sensors that only return 0 or 1
                boolean active = (distance >= 0.0 && distance < PROXIMITY_THRESHOLD &&
                        distance < mProximitySensor.getMaximumRange());   //如果距离小于某一个距离阈
值，默认是 5.0f，说明手机和脸部距离贴近，应该要熄灭屏幕

                if (mDebugProximitySensor) {
                    Slog.d(TAG, "mProximityListener.onSensorChanged active: " + active);
                }
                if (timeSinceLastEvent < PROXIMITY_SENSOR_DELAY) {
                    // enforce delaying atleast PROXIMITY_SENSOR_DELAY before processing
                    mProximityPendingValue = (active ? 1 : 0);
                    mHandler.postDelayed(mProximityTask, PROXIMITY_SENSOR_DELAY - timeSinceLast -
Event);
                    proximityTaskQueued = true;
                } else {
                    // process the value immediately
                    mProximityPendingValue = -1;
                    proximityChangedLocked(active);    //熄灭屏幕操作
```

```
                }

                // update mProximityPartialLock state
                boolean held = mProximityPartialLock.isHeld();
                if (!held && proximityTaskQueued) {
                    // hold wakelock until mProximityTask runs
                    mProximityPartialLock.acquire();
                } else if (held && !proximityTaskQueued) {
                    mProximityPartialLock.release();
                }
            }
        }

        public void onAccuracyChanged(Sensor sensor, int accuracy) {
            // ignore
        }
    };
```

由上述代码可知，在监听时会首先通过“float distance = event.values[0];”获取变化的距离。如果发现检测这次距离变化和上次距离变化时间差，例如小于系统设置的阈值则不会去熄灭屏幕。过于频繁的操作系统会忽略掉。如果感觉 P-Sensor 不够灵敏，可以修改如下所示的系统默认值。

```
private static final int PROXIMITY_SENSOR_DELAY = 1000;
```

将上述值改小后就会发现 P-Sensor 会变得灵敏很多。

如果 P-Sensor 检测到这次距离变化小于系统默认值，并且这次是一次正常的变化，那么需要通过如下代码熄灭屏幕。

```
proximityChangedLocked(active);
```

此处会判断 P-Sensor 是否可以用，如果不可用则返回，并忽略这次距离变化。

```
if (!mProximitySensorEnabled) {
            Slog.d(TAG, "Ignoring proximity change after sensor is disabled");
            return;
        }
```

如果一切都满足，则调用如下代码灭灯。

```
goToSleepLocked(SystemClock.uptimeMillis(),
                    WindowManagerPolicy.OFF_BECAUSE_OF_PROX_SENSOR);
```

5.9　气压传感器详解

知识点讲解：光盘:视频\知识点\第 5 章\气压传感器详解.avi

在 Android 设备开发应用过程中，通常需要使用设备来感知当前所处环境的信息，例如气压、GPS、海拔、湿度和温度。在 Android 系统中，专门提供了气压传感器、海拔传感器、湿度传感器和温度传感器来支持上述功能。本节将详细讲解在 Android 设备中使用气压传感器详解的基本知识，为读者学习本书后面的知识打下基础。

5.9.1 气压传感器基础

在现实应用中，气压传感器主要用于测量气体的绝对压强，主要适用于与气体压强相关的物理实验，如气体定律等，也可以在生物和化学实验中测量干燥、无腐蚀性的气体压强。气压传感器的原理比较简单，其主要的传感元件是一个对气压传感器内的强弱敏感的薄膜和一个顶针开关控制，电路方面它连接了一个柔性电阻器。当被测气体的压强降低或升高时，这个薄膜变形带动顶针，同时该电阻器的阻值将会改变。从传感元件取得 0～5V 的信号电压，经过 A/D 转换由数据采集器接收，然后数据采集器以适当的形式把结果传送给计算机。

在现实应用中，很多气压传感器的主要部件为变容式硅膜盒。当该变容硅膜盒受外界大气压力发生变化时顶针动作，单晶硅膜盒随着发生弹性变形，从而引起硅膜盒平行板电容器电容量的变化来控制气压传感器。

国标 GB7665—87 对传感器的定义是："能感受规定的被测量并按照一定的规律转换成可用信号的器件或装置，通常由敏感元件和转换元件组成"。而气压传感器是由一种检测装置，能感受到被测量的信息，并能将检测感受到的信息，按一定规律变换成为电信号或其他所需形式的信息输出，以满足信息的传输、处理、存储、显示、记录和控制等要求，是实现自动化检测和控制的首要环节。

5.9.2 气压传感器在智能手机中的应用

随着智能手机设备的发展，气压传感器得到了大力的普及。气压传感器首次在智能手机上使用是在 Galaxy Nexus 上，而之后推出的一些 Android 旗舰手机里也包含了这一传感器，像 Galaxy SIII、Galaxy Note 2 也都有。对于喜欢登山的人来说，都会非常关心自己所处的高度。海拔高度的测量方法，一般常用的有两种方式，一是通过 GPS 全球定位系统，二是通过测出大气压，然后根据气压值计算出海拔高度。由于受到技术和其他方面原因的限制，GPS 计算海拔高度一般误差都会有十米左右，而如果在树林里或者是在悬崖下面时，有时候甚至接收不到 GPS 卫星信号。同时当用户处于楼宇内时，内置感应器可能会无法接收到 GPS 信号，从而不能够识别地理位置。配合气压传感器、加速计、陀螺仪等就能够实现精确定位。这样当你在商场购物时，你能够更好找到目标商品。

另外在汽车导航领域中，经常会有人抱怨在高架桥里导航常常会出错。比如在高架桥上时，GPS 说右转，而实际上右边根本没有右转出口，这主要是 GPS 无法判断你是桥上还是桥下而造成的错误导航。一般高架桥上下两层的高度都会有几米到十几米的距离，而 GPS 的误差可能会有几十米，所以发生错误的导航也就可以理解了。此时如果在手机中增加一个气压传感器就不一样了，它的精度可以做到 1 米的误差，这样就可以很好地辅助 GPS 来测量出所处的高度，错误导航的问题也就容易解决了。

而气压的方式可选择的范围会广些，而且可以把成本可以控制在比较低的水平。另外像 Galaxy Nexus 等手机的气压传感器还包括温度传感器，它可以捕捉到温度来对结果进行修正，以增加测量结果的精度。所以在手机原有 GPS 的基础上再增加气压传感器的功能，可以让三维定位更加精准。

在 Android 系统中，气压传感器的类型是 TYPE_PRESSURE，单位是 hPa（百帕斯卡），能够返回当前环境下的压强。

5.10　温度传感器基础

知识点讲解：光盘:视频\知识点\第 5 章\温度传感器基础.avi

温度传感器从 17 世纪初人们开始利用温度进行测量。在半导体技术的支持下，现在相继开发了半导体热电偶传感器、PN 结温度传感器和集成温度传感器。与之相应，根据波与物质的相互作用规律，相继开发了声学温度传感器、红外传感器和微波传感器。温度传感器是五花八门的各种传感器中最为常用的一种，现代的温度传感器外形非常小，这样更加让它广泛应用在生产实践的各个领域中，也为人们的生活提供了无数的便利。

在现实应用中，温度传感器有 4 种主要类型：热电偶、热敏电阻、电阻温度检测器（RTD）和 IC 温度传感器。IC 温度传感器又包括模拟输出和数字输出两种类型。在现实世界中，温度传感器是温度测量仪表的核心部分，品种繁多。按测量方式可以分为接触式和非接触式两大类，按照传感器材料及电子元件特性分为热电阻和热电偶两类。

在当前的技术水平条件下，温度传感器的主要原理如下。

（1）金属膨胀原理设计的传感器

金属在环境温度变化后会产生一个相应的延伸，因此传感器可以以不同方式对这种反应进行信号转换。

（2）双金属片式传感器

双金属片由两片不同膨胀系数的金属贴在一起而组成，随着温度变化，材料 A 比另外一种金属膨胀程度要高，引起金属片弯曲。弯曲的曲率可以转换成一个输出信号。

（3）双金属杆和金属管传感器

随着温度升高，金属管（材料 A）长度增加，而不膨胀钢杆（金属 B）的长度并不增加，这样由于位置的改变，金属管的线性膨胀就可以进行传递。反过来，这种线性膨胀可以转换成一个输出信号。

（4）液体和气体的变形曲线设计的传感器

在温度变化时，液体和气体同样会相应产生体积的变化。

综上所述，多种类型的结构可以把这种膨胀的变化转换成位置的变化，这样产生位置的变化可以输出为电位计、感应偏差、挡流板等形式的结果。

在 Android 系统中，早期版本的温度传感器值是 TYPE_TEMPERATURE，在新版本中被 TYPE_AMBIENT_TEMPERATURE 替换。Android 温度传感器的单位是℃，能够测量并返回当前的温度。

在 Android 内核的平台中自带了大量的传感器源码，读者可以在 Rexsee 的开源社区 http://www.rexsee.com/找到相关的源代码。

5.11　湿度传感器基础

知识点讲解：光盘:视频\知识点\第 5 章\湿度传感器基础.avi

人类的生存和社会活动与湿度密切相关。随着现代化的实现，很难找出一个与湿度无关的领域来。由于应用领域不同，对湿度传感器的技术要求也不同。

在 Android 系统中，湿度传感器的值是 TYPE_RELATIVE_HUMIDITY，单位是%，能够测量周围环境的相对湿度。Android 系统中的湿度与光线、气压、温度传感器的使用方式相同，可以从湿度传感器中读取到相对湿度的原始数据。而且，如果设备同时提供了湿度传感器（TYPE_RELATIVE_ HUMIDITY）和温度传感器（TYPE_AMBIENT_TEMPERATURE），那么就可以用这两个数据流来计算出结露点和绝对湿度。

（1）结露点

结露点是在固定的气压下，空气中所含的气态水达到饱和而凝结成液态水所需要降至的温度。以下给出了计算结露点温度的公式：

$$t_d(t,RH) = T_n \cdot \frac{\ln(RH/100\%) + m \cdot t/(T_n+t)}{m - [\ln(RH/100\%) + m \cdot t/(T_n+t)]}$$

在上述公式中，各个参数的具体说明如下。

- t_d = 结露点温度，单位是℃。
- t = 当前温度，单位是℃。
- RH = 当前相对湿度，单位是百分比（%）。
- m = 18.62。
- T_n = 243.12。

（2）绝对湿度

绝对湿度是在一定体积的干燥空气中含有的水蒸气的质量。绝对湿度的计量单位是 g/m^3。以下给出了计算绝对湿度的公式：

$$d_v(t,RH) = 218.7 \cdot \frac{(RH/100\%) \cdot A \cdot \exp(m \cdot t/(T_n+t))}{273.15 + t}$$

在上述公式中，各个参数的具体说明如下。

- d_v = 绝对湿度，单位是 g/m^3。
- t = 当前温度，单位是℃。
- RH = 当前相对湿度，单位是百分比（%）。
- m = 18.62。
- T_n = 243.12℃。
- A = 6.112 hPa。

第 6 章　蓝牙系统详解

蓝牙这一名称来自于 10 世纪的一位丹麦国王 Harald Blatand，Blatand 在英文里的意思可以被解释为 Bluetooth。因为国王喜欢吃蓝莓，牙龈每天都是蓝色的所以叫蓝牙。蓝牙的创始人是瑞典爱立信公司，爱立信早在 1994 年就已进行研发。1997 年，爱立信与其他设备生产商联系，并激发了他们对该项技术的浓厚兴趣。1998 年 2 月，5 个跨国大公司，包括爱立信、诺基亚、IBM、东芝及 Intel 组成了一个特殊兴趣小组（SIG），他们共同的目标是建立一个全球性的小范围无线通信技术，也就是现在的蓝牙。在 Android 设备中，蓝牙技术是常用的一种近距离数据传输技术。本章将详细讲解 Android 蓝牙系统的基本架构知识。

6.1　短距离无线通信技术概览

知识点讲解：光盘:视频\知识点\第 6 章\短距离无线通信技术概览.avi

在 Android 外设项目开发应用中，需要建立外部设备与控制设备的连接，此时发挥关键作用的是短距离无线传输技术。目前有多种短距离无线传输技术可以应用在外设项目开发过程中，在现实中除了已经得到大规模应用的 RFID 之外，还有 WiFi、ZigBee、蓝牙等比较成熟的技术，以及基于这些技术发展而来的新技术。这些技术各具特点，因对其传输速度、距离、耗电量等方面的要求不同，形成了各自不同的外设项目开发过程应用场景。本节将简要介绍当今可以实现短距离无线通信功能的常用技术。

6.1.1　ZigBee

ZigBee 以其鲜明的技术特点在外设项目开发过程中受到了高度关注，该技术使用的频段分别为 2.4GHz、868MHz（欧洲）及 915MHz（美国）。其主要的技术特点：一是数据传输速率低，只有 10Kbps～250Kbps；二是功耗低，低传输速率带来了仅为 1 毫瓦的低发射功率。据估算，ZigBee 设备仅靠两节 5 号电池就可以维持长达 6 个月到两年左右的使用时间，这是 ZigBee 的一个独特优势；三是成本低，因为 ZigBee 传输速率低、协议简单；四是网络容量大，每个 ZigBee 网络最多可以支持 255 个设备，一个区域内可以同时存在最多 100 个 ZigBee 网络，网络组成灵活。ZigBee 芯片主要企业有德州仪器、飞思卡尔等。市场调研机构 ABI Reserch 的一份数据显示，2005—2012 年，ZigBee 市场的年均复合增长率为 63%。

ZigBee 是从家庭自动化开始的，在瑞典哥德堡就是从智能电表开始，然后进一步用到燃气表、水表、热力表等家庭各种计量表。在 2011 年中国无线世界暨外设项目开发过程大会上，ZigBee 联盟大中华区代表黄家瑞说，“ZigBee 在智能电表里不仅仅是远程抄表工具，它是一个终端，也是一个网关，这些网关结合在一起，整个小区就变成了智能电网小区，智能电表可以搜集家里所有家电的用电信息。”

目前，ZigBee 正在完善其网关标准，2011 年 7 月底发布了第十个标准 ZigBee Gateway（ZigBee 网关）。ZigBee Gateway 提供了一种简单、高成本效益的互联网连接方式，使服务提供商、企业和个人消费者有机会运行这些设备并将 ZigBee 网络连接至互联网。ZigBee Gateway 是 ZigBee Network Devicesp（ZigBee 网络设备）这一新类别范畴的首个标准，这将使 ZigBee 发展进一步提速。

6.1.2 WiFi

WiFi 是以太网的一种无线扩展技术，如果有多个用户同时通过一个热点接入，带宽将被这些用户共享，WiFi 的速率会降低，处于 2.4GHz 频段的 WiFi 信号受墙壁阻隔的影响较小。WiFi 的传输速率随着技术的演进还在不断提高，我国电信运营商在构建无线城市中采用的 WiFi 技术部分已经升级到 802.11n，最高速率从 802.11g 标准的 11Mbps 提高到 50Mbps 以上。在 WiFi 产业链中，最大的芯片企业是博通。

在笔记本电脑和手机上已经得到广泛应用的 WiFi 正在向消费电子产品渗透，Myron Hattig 说：“除了手机外，已经有 25%的消费类电子设备使用 WiFi，在打印机、洗衣机上都在使用 WiFi，家用电器生产商协会将 WiFi 作为一个更高级别的智能电器沟通技术。WiFi 可以将设备与设备相连，从而使整个家庭的家用电器、电子设备相连。”

最大 WiFi 芯片制造商博通正在推动 WiFi Direct 标准的商用，以支持这种设备到设备的直连。特别是在家庭互联中，相片、视频等大数据量的业务在手机、平板电脑、电视等设备中的直连应用前景广阔。

基于 WiFi 上发展起来的 WIGIG 也是未来家庭互联市场有力的竞争技术。该技术可工作在 40～60GHz 的超高频段，其传输速度可以达到 1Gbps 以上，不能穿过墙壁。目前英特尔、高通等芯片企业在支持 WIGIG 发展，该技术还在完善中，如需要进一步降低功耗等。

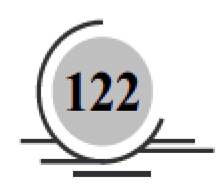
6.1.3 蓝牙

使用“蓝牙”（Bluetooth）技术可以有效地简化移动通信终端设备之间的通信，也能够成功地简化设备与因特网（Internet）之间的通信，从而使数据传输变得更加迅速高效，为无线通信拓宽道路。蓝牙采用分散式网络结构以及快跳频和短包技术，支持点对点及点对多点通信，工作在全球通用的 2.4GHz ISM（即工业、科学、医学）频段。蓝牙技术的数据传输速率为 1Mbps，采用时分双工传输方案实现全双工传输。

蓝牙是一种无线传输技术，许多行业的制造商都积极地在其产品中实施此技术，以减少使用零乱的电线，实现无缝连接、流传输立体声，传输数据或进行语音通信。Bluetooth 技术在 2.4 GHz 波段运行，该波段是一种无须申请许可证的工业、科技、医学（ISM）无线电波段。正因如此，使用 Bluetooth 技术不需要支付任何费用。但必须向手机提供商注册使用 GSM 或 CDMA，除了设备费用外，不需要为使用 Bluetooth 技术再支付任何费用。

Bluetooth 技术得到了空前广泛的应用，集成该技术的产品从手机、汽车到医疗设备，使用该技术的用户从消费者、工业市场到企业等，不一而足。低功耗、小体积以及低成本的芯片解决方案使得 Bluetooth 技术甚至可以应用于极微小的设备中。

Bluetooth 技术是一项即时技术，它不要求固定的基础设施，且易于安装和设置，不需要电缆即可



实现连接。新用户使用亦不费力，我们只需拥有 Bluetooth 品牌产品，检查可用的配置文件，将其连接至使用同一配置文件的另一 Bluetooth 设备即可。后续的 PIN 码流程就如同在 ATM 机器上操作一样简单。外出时，可以随身带上个人局域网（PAN），甚至可以与其他网络连接。

蓝牙可以在包括移动电话、PDA、无线耳机、笔记本电脑、相关外设等众多设备之间进行无线信息交换。蓝牙采用分散式网络结构以及快跳频和短包技术，支持点对点及点对多点通信，工作在全球通用的 2.4GHz 频段，其数据速率为 1Mbps。

2010 年 7 月，以低功耗为特点的蓝牙 4.0 标准推出，蓝牙大中华区技术市场经理吕荣良将其看作蓝牙第二波发展高潮的标志，他表示："蓝牙可以跨领域应用，主要有 4 个生态系统，分别是智能手机与笔记本电脑等终端市场、消费电子市场、汽车前装市场和健身运动器材市场。"

NFC 和 UWB 曾经是十分受关注的短距离无线接入技术，但其发展已经日渐势微。业内专家认为，无线频谱的规划和利用在短距离通信中日益重要。短距离通信技术目前主要采用 2.4GHz 的开放频谱，但随着外设项目开发过程的发展和大量短距离通信技术的应用，频谱需求会快速增长，视频、图像等大数据量的通信正在寻求更高频段的解决方案。

6.1.4　NFC

NFC 是近场通信（Near Field Communication）的缩写，此技术由非接触式射频识别（RFID）演变而来，由飞利浦半导体（现恩智浦半导体）、诺基亚和索尼共同研制开发，其基础是 RFID 及互连技术。NFC 是一种短距高频的无线电技术，在 13.56MHz 频率运行于 20 厘米距离内。其传输速度有 106 Kbit/s、212 Kbit/s 或者 424 Kbit/s 3 种，已成为 ISO/IEC IS 18092 国际标准、ECMA-340 标准与 ETSI TS 102 190 标准。NFC 采用主动和被动两种读取模式。

NFC 近场通信技术是由非接触式射频识别（RFID）及互联互通技术整合演变而来，在单一芯片上结合感应式读卡器、感应式卡片和点对点的功能，能在短距离内与兼容设备进行识别和数据交换。工作频率为 13.56MHz，但是使用这种手机支付方案的用户必须更换特制的手机。目前这项技术在日韩被广泛应用。手机用户凭着配置了支付功能的手机就可以行遍全国：他们的手机可以用作机场登机验证、大厦的门禁钥匙、交通一卡通、信用卡、支付卡等。

NFC 和蓝牙都是短程通信技术，而且都被集成到移动电话，但 NFC 不需要复杂的设置程序。NFC 也可以简化蓝牙连接。NFC 略胜蓝牙的地方在于设置程序较短，但无法达到低功率蓝牙（Bluetooth Low Energy）的速度。在两台 NFC 设备相互连接的设备识别过程中，使用 NFC 来替代人工设置会使创建连接的速度大大加快，会少于十分之一秒。

6.2　低功耗蓝牙基础

知识点讲解：光盘:视频\知识点\第 6 章\低功耗蓝牙基础.avi

BLE 是 Bluetooth Low Energy 的缩写，意为低功耗蓝牙，是对传统蓝牙 BR/EDR 技术的补充。尽管 BLE 和传统蓝牙都被称为蓝牙标准，并且都共享射频，但是 BLE 是一个完全不一样的技术。BLE 不具备和传统蓝牙 BR/EDR 的兼容性，是专为小数据率、离散传输的应用而设计的。本节将详细讲解低功耗蓝牙技术的基本知识。

6.2.1 低功耗蓝牙的架构

BLE 协议架构总体上分成 3 层，从下到上分别是：控制器（Controller）、主机（Host）和应用端（Apps）。三者可以在同一芯片类实现，也可以分不同芯片内实现，控制器（Controller）是处理射频数据解析、接收和发送，主机（Host）是控制不同设备之间如何进行数据交换；应用端（Apps）实现具体应用。

（1）控制器 Controller

Controller 实现射频相关的模拟和数字部分，完成最基本的数据发送和接收，Controller 对外接口是天线，对内接口是主机控制器接口 HCI（Host Controller Interface）；控制器包含物理层 PHY（Physical Layer）、链路层 LL（Linker Layer）、直接测试模式 DTM（Direct Test Mode）以及主机控制器接口 HCI。

❑ 物理层PHY

GFSK 信号调制，2402MHz~2480MHz，40 个 channel，每两个 channel 间隔 2MHz（经典蓝牙协议是 1MHz），数据传输速率是 1Mbps。

❑ 直接测试模式DTM

为射频物理层测试接口，射频数据分析之用。

❑ 链路层LL

基于物理层 PHY 之上，实现数据通道分发、状态切换、数据包校验和加密等；链路层 LL 分 2 种通道：广播通道（advertising channels）和数据通道（data channels）；广播通道有 3 个，37ch（2402MHz），38ch（2426MHz），39ch（2480MHz），每次广播都会往这 3 个通道同时发送（并不会在这 3 个通道之间跳频），为防止某个通道被其他设备阻塞，以至于设备无法配对或广播数据，之所以定 3 个广播通道是一种权衡，少了可能会被阻塞，多了加大功耗，还有一个有意思的事情是，3 个广播通道刚好避开了 WiFi 的 1ch、6ch、11ch，所以在 BLE 广播的时候，不至于被 WiFi 影响；当 BLE 匹配之后，链路层 LL 由广播通道切换到数据通道，数据通道 37 个，数据传输的时候会在这 37 个通道间切换，切换规则在设备间匹配时候约定。

（2）主机 Host/控制器 Controller 接口 HCI

HCI 作为一种接口，存在于主机 Host 和控制器 Controller 中，控制器 Host 通过 HCI 发送数据和事件给主机，主机 Host 通过 HCI 发送命令和数据给控制器 Controller. HCI 逻辑上定义一系列的命令、事件；物理上有 UART、SDIO、USB，实际可能包含里面的任意一种或几种。

6.2.2 低功耗蓝牙分类

BLE 通常应用在传感器和智能手机或者平板的通信中。到目前为止，只有很少的智能机和平板支持 BLE，如 iPhone 4S 以后的苹果手机、Motorola Razr 和 the new iPad 及其以后的 iPad。安卓手机也逐渐支持 BLE，安卓的 BLE 标准在 2013 年 7 月 24 日刚发布。智能机和平板会带双模蓝牙的基带和协议栈，协议栈中包括 GATT 及以下的所有部分，但是没有 GATT 之上的具体协议。所以，这些具体的协议需要在应用程序中实现，实现时需要基于各个 GATT API 集。这样有利于在智能机端简单地实现具体协议，也可以在智能机端简单地开发出一套基于 GATT 的私有协议。

在现实应用中，低功耗蓝牙分为单模（Bluetooth Smart）和双模（Bluetooth Smart Ready）两种设备。BLE 和蓝牙 BR/EDR 有所区分，这样可以让我们用 3 种方式将蓝牙技术集成到具体设备中。因为

不再是所有现有的蓝牙设备可以和另一个蓝牙设备进行互联，所以准确描述产品中蓝牙的版本是非常重要的。在接下来的内容中，将详细讲解单模蓝牙和双模蓝牙的基本知识。

（1）单模蓝牙

单模蓝牙设备被称为 Bluetooth Smart 设备，并且有专用的 Logo，如图 6-1 所示。

在现实应用中，手表、运动传感器等小型设备通常是基于低功耗单模蓝牙的。为了实现极低的功耗效果，在硬件和软件上都进行了优化，这样的设备只能支持 BLE。单模蓝牙芯片往往是一个带有单模蓝牙协议栈的产品，这个协议栈通常是芯片商免费提供的。

（2）双模蓝牙

双模蓝牙设备被称为 Bluetooth Smart Ready 设备，并且有专用的 Logo，如图 6-2 所示。



图 6-1　Bluetooth Smart 设备



图 6-2　Bluetooth Smart Ready 设备

双模设备支持蓝牙 BR/EDR 和 BLE。在双模设备中，BR/EDR 和 BLE 技术使用同一个射频前端和天线。典型的双模设备有智能手机、平板电脑、PC 和 Gateway。这些设备可以接收到通过 BLE 或者蓝牙 BR/EDR 设备发送过来的数据，这些设备往往都有足够的供电能力。双模设备和 BLE 设备通信的功耗低于双模设备和蓝牙 BR/EDR 设备通信的功耗。在使用双模解决方案时，需要用一个外部处理器才可以实现蓝牙协议栈。

6.2.3　BLE 和传统蓝牙 BR/EDR 技术的对比

BLE 和传统蓝牙 BR/EDR 技术相比的细节如表 6-1 所示。

表 6-1　BLE 和传统蓝牙 BR/EDR 技术的对比

对　比	Bluetooth BR/EDR	Bluetooth Low Energy
Frequency	2400～2483.5 MHz	2400～2483.5 MHz
Deep Sleep	~80 μA	<5 μA
Idle	~8 mA	~1 mA
Peak Current	6～40 mA	10～30 mA
Range	500m (Class 1) / 50m (Class 2)	100m
Min. Output Power	0 dBm (Class 1) / −6 dBm (Class 2)	−20 dBm
Max. Output Power	+20 dBm (Class 1) / +4 dBm (Class 2)	+10 dBm
Receiver Sensitivity	≥ −70 dBm	≥ −70 dBm
Encryption	64 bit / 128 bit	AES−128 bit
Connection Time	100 ms	3 ms
Frequency Hopping	Yes	Yes
Advertising Channel	32	3
Data Channel	79	37
Voice Capable	Yes	No

6.3　Android 系统中的蓝牙模块

知识点讲解：光盘:视频\知识点\第 6 章\Android 系统中的蓝牙模块.avi

Android 平台的蓝牙系统是基于 BlueZ 实现的，是通过 Linux 中一套完整的蓝牙协议栈开源实现的。当前 BlueZ 被广泛应用于各种 Linux 版本中，并被芯片公司移植到各种芯片平台上所使用。在 Linux 2.6 内核中已经包含了完整的 BlueZ 协议栈，在 Android 系统中已经移植并嵌入进了 BlueZ 的用户空间实现，并且随着硬件技术的发展而不断更新。

蓝牙（Bluetooth）技术实际上是一种短距离无线电技术。在 Android 系统的蓝牙模块中，除了使用 Kernel 支持外，还需要用户空间的 BlueZ 的支持。

Android 平台中蓝牙模块的基本层次结构如图 6-3 所示。



图 6-3　蓝牙系统的层次结构

Android 平台中蓝牙系统从上到下主要包括 Java 框架中的 BlueTooth 类、Android 适配库、BlueZ 库、驱动程序和协议，这几部分的系统结构如图 6-4 所示。

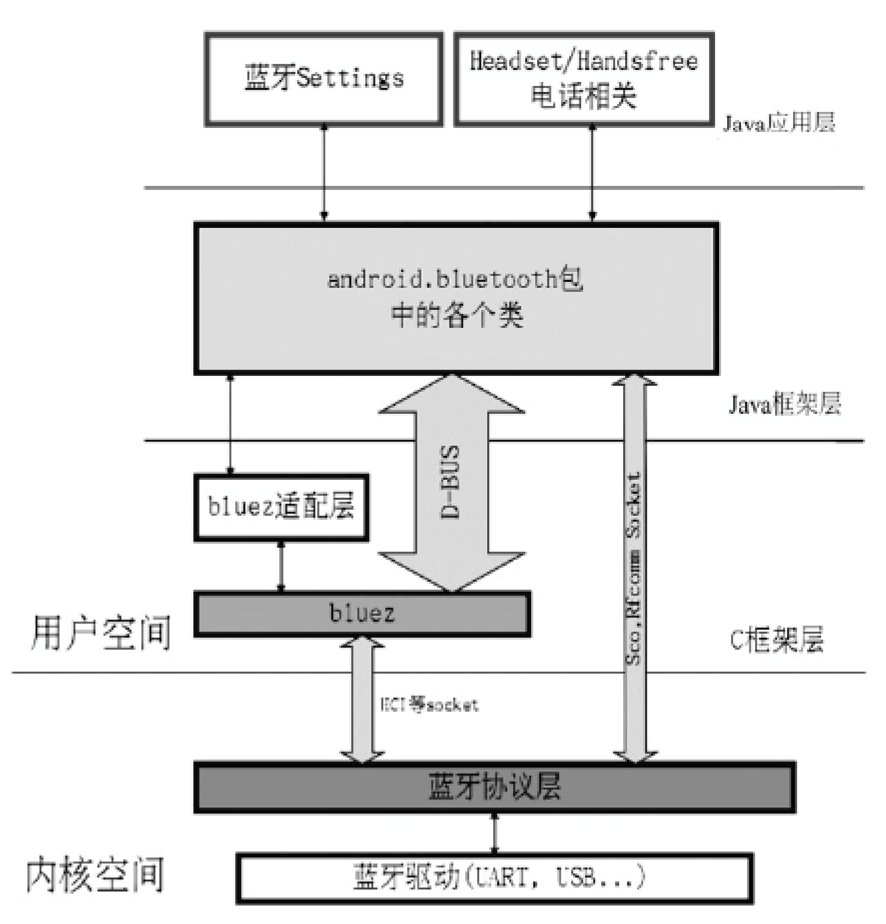
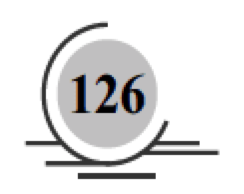


图 6-4　蓝牙系统结构



在图 6-4 中各个层次结构的具体说明如下。

（1）BlueZ 库

Android 蓝牙设备管理的库的路径如下所示。

```
external/bluez/
```

可以分别生成库 libbluetooth.so、libbluedroid.so 和 hcidump 等众多相关工具和库。BlueZ 库提供了对用户空间蓝牙的支持，在里面包含了主机控制协议 HCI 以及其他众多内核实现协议的接口，并且实现了所有蓝牙应用模式 Profile。

（2）蓝牙的 JNI 部分

此部分的代码路径如下所示。

```
frameworks/base/core/jni/
```

（3）Java 框架层

Java 框架层的实现代码保存在如下路径。

```
frameworks/base/core/java/android/bluetooth                //蓝牙部分对应应用程序的 API
frameworks/base/core/java/android/Server                   //蓝牙的服务部分
```

蓝牙的服务部分负责管理并使用底层本地服务，并封装成系统服务。而在 android.bluetooth 部分中包含了各个蓝牙平台的 API 部分，以供应用程序层所使用。

（4）BlueTooth 的适配库

BlueTooth 适配库的代码路径如下所示。

```
system/bluetooth/
```

在此层用于生成库 libbluedroid.so 以及相关工具和库，能够实现对蓝牙设备的管理，例如蓝牙设备的电源管理。

6.4　和蓝牙相关的类

知识点讲解：光盘:视频\知识点\第 6 章\和蓝牙相关的类.avi

经过本章前面内容的学习，读者已经了解了 Android 系统中蓝牙的基本知识；根据对上述从底层到应用的学习，了解了蓝牙的工作原理和机制。本节将详细讲解在 Android 系统中和蓝牙相关的类，为读者学习本书后面的知识打好基础。

6.4.1　BluetoothSocket 类

1. BluetoothSocket 类基础

BluetoothSocket 类的定义格式如下所示。

```
public static class Gallery.LayoutParams extends ViewGroup.LayoutParams
```

BluetoothSocket 类的定义结构如下所示。

```
java.lang.Object
android.view. ViewGroup.LayoutParams
android.widget.Gallery.LayoutParams
```

Android 的蓝牙系统和 Socket 套接字密切相关，蓝牙端的监听接口和 TCP 的端口类似，都是使用了 Socket 和 ServerSocket 类。在服务器端，使用 BluetoothServerSocket 类来创建一个监听服务端口。

当一个连接被 BluetoothServerSocket 所接受，它会返回一个新的 BluetoothSocket 来管理该连接。在客户端，使用一个单独的 BluetoothSocket 类去初始化一个外接连接和管理该连接。

最通常使用的蓝牙端口是 RFCOMM，它是被 Android API 支持的类型。RFCOMM 是一个面向连接，通过蓝牙模块进行的数据流传输方式，也被称为串行端口规范（Serial Port Profile，SPP）。

为了创建一个 BluetoothSocket 去连接到一个已知设备，使用方法 BluetoothDevice.createRfcommSocketToServiceRecord()。然后调用 connect()方法去尝试一个面向远程设备的连接。这个调用将被阻塞直到一个连接已经建立或者该连接失效。

为了创建一个 BluetoothSocket 作为服务端（或者“主机”），每当该端口连接成功后，无论它初始化为客户端，或者被接受作为服务器端，都通过方法 getInputStream()和 getOutputStream()来打开 I/O 流，从而获得各自的 InputStream 和 OutputStream 对象。

BluetoothSocket 类的线程是安全的，因为 close()方法总会马上放弃外界操作并关闭服务器端口。

2. BluetoothSocket 类的公共方法

（1）public void close ()

功能：马上关闭该端口并且释放所有相关的资源。在其他线程的该端口中引起阻塞，从而使系统马上抛出一个 I/O 异常。

异常：IOException。

（2）public void connect ()

功能：尝试连接到远程设备。该方法将阻塞，指导一个连接建立或者失效。如果该方法没有返回异常值，则该端口现在已经建立。当设备查找正在进行的时候，创建对远程蓝牙设备的新连接不可被尝试。设备查找在蓝牙适配器上是一个重量级过程，并且肯定会降低一个设备的连接。使用 cancelDiscovery()方法会取消一个外界的查询，因为这个查询并不由活动所管理，而是作为一个系统服务来运行，所以即使它不能直接请求一个查询，应用程序也总会调用 cancelDiscovery()方法。使用方法 close()可以用来放弃从另一线程而来的调用。

异常：IOException，表示一个错误，例如连接失败。

（3）public InputStream getInputStream ()

功能：通过连接的端口获得输入数据流。即使该端口未连接，该输入数据流也会返回。不过在该数据流上的操作将抛出异常，直到相关的连接已经建立。

返回值：输入流。

异常：IOException。

（4）public OutputStream getOutputStream ()

功能：通过连接的端口获得输出数据流。即使该端口未连接，该输出数据流也会返回。不过在该数据流上的操作将抛出异常，直到相关的连接已经建立。

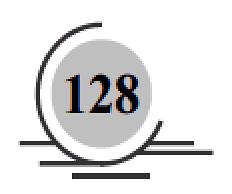
返回值：输出流。

异常：IOException。

（5）public BluetoothDevice getRemoteDevice ()

功能：获得该端口正在连接或者已经连接的远程设备。

返回值：远程设备。



6.4.2　BluetoothServerSocket 类

1. BluetoothServerSocket 类基础

BluetoothServerSocket 类的格式如下所示。

```
public final class BluetoothServerSocket extends Object implements Closeable
```

BluetoothServerSocket 类的结构如下所示。

```
java.lang.Object
android.bluetooth.BluetoothServerSocket
```

2. BluetoothServerSocket 类的公共方法

（1）public BluetoothSocketaccept (int timeout)

功能：阻塞直到超时时间内的连接建立。在一个成功建立的连接上返回一个已连接的 BluetoothSocket 类。每当该调用返回的时候，它可以在此调用去接收以后新来的连接。close()方法可以用来放弃从另一线程来的调用。

参数 timeout：表示阻塞超时时间。

返回值：已连接的 BluetoothSocket。

异常：IOException，表示出现错误，比如该调用被放弃或超时。

（2）public BluetoothSocket accept ()

功能：阻塞直到一个连接已经建立。在一个成功建立的连接上返回一个已连接的 BluetoothSocket 类。每当该调用返回的时候，它可以在此调用去接收以后新来的连接。使用 close()方法可以用来放弃从另一线程来的调用。

返回值：已连接的 BluetoothSocket。

异常：IOException，表示出现错误，比如该调用被放弃或者超时。

（3）public void close ()

功能：马上关闭端口，并释放所有相关的资源。在其他线程的该端口中引起阻塞，从而使系统马上抛出一个 I/O 异常。关闭 BluetoothServerSocket 不会关闭接受自 accept()的任意 BluetoothSocket。

异常：IOException。

6.4.3　BluetoothAdapter 类

1. BluetoothAdapter 类基础

BluetoothAdapter 类的格式如下所示。

```
public final class BluetoothAdapter extends Object
```

BluetoothAdapter 类的结构如下所示。

```
java.lang.Object
android.bluetooth.BluetoothAdapter
```

BluetoothAdapter 代表本地的蓝牙适配器设备，通过此类可以让用户能执行基本的蓝牙任务。例如初始化设备的搜索，查询可匹配的设备集，使用一个已知的 MAC 地址来初始化一个 BluetoothDevice 类，创建一个 BluetoothServerSocket 类以监听其他设备对本机的连接请求等。

为了得到这个代表本地蓝牙适配器的 BluetoothAdapter 类，需要调用静态方法 getDefaultAdapter()，这是所有蓝牙动作使用的第一步。当拥有本地适配器以后，用户可以获得一系列的 BluetoothDevice 对象，这些对象代表所有拥有 getBondedDevice()方法的已经匹配的设备；用 startDiscovery()方法来开始设备的搜寻；或者创建一个 BluetoothServerSocket 类，通过 listenUsingRfcommWithServiceRecord(String, UUID)方法来监听新来的连接请求。

注意：大部分方法需要BLUETOOTH权限，一些方法同时需要BLUETOOTH_ADMIN权限。

2. BluetoothAdapter 类的常量

（1）String ACTION_DISCOVERY_FINISHED

广播事件：本地蓝牙适配器已经完成设备的搜寻过程，需要 BLUETOOTH 权限接收。

常量值：android.bluetooth.adapter.action.DISCOVERY_FINISHED。

（2）String ACTION_DISCOVERY_STARTED

广播事件：本地蓝牙适配器已经开始对远程设备的搜寻过程。它通常涉及一个大概需时 12 秒的查询扫描过程，紧跟着是一个对每个获取到自身蓝牙名称的新设备的页面扫描。用户会发现一个把 ACTION_FOUND 常量通知为远程蓝牙设备的注册。设备查找是一个重量级过程。当查找正在进行的时候，用户不能尝试对新的远程蓝牙设备进行连接，同时存在的连接将获得有限制的带宽以及高等待时间。用户可用 cancelDiscovery()方法来取消正在执行的查找进程，需要 BLUETOOTH 权限接收。

常量值：android.bluetooth.adapter.action.DISCOVERY_STARTED。

（3）String ACTION_LOCAL_NAME_CHANGED

广播活动：本地蓝牙适配器已经更改了它的蓝牙名称。该名称对远程蓝牙设备是可见的，它总是包含一个带有名称的 EXTRA_LOCAL_NAME 附加域，需要 BLUETOOTH 权限接收。

常量值：android.bluetooth.adapter.action.LOCAL_NAME_CHANGED。

（4）String ACTION_REQUEST_DISCOVERABLE

Activity 活动：显示一个请求被搜寻模式的系统活动。如果蓝牙模块当前未打开，该活动也将请求用户打开蓝牙模块。被搜寻模式和 SCAN_MODE_CONNECTABLE_DISCOVERABLE 等价。当远程设备执行查找进程的时候，它允许其发现该蓝牙适配器。从隐私安全考虑，Android 不会将被搜寻模式设置为默认状态。该意图的发送者可以选择性地运用 EXTRA_DISCOVERABLE_DURATION 这个附加域去请求发现设备的持续时间。普遍来说，对于每一请求，默认的持续时间为 120 秒，最大值则可达到 300 秒。

Android 运用 onActivityResult(int, int, Intent)回收方法来传递该活动结果的通知。被搜寻的时间（以秒为单位）将通过 resultCode 值来显示，如果用户拒绝被搜寻，或者设备产生了错误，则通过 RESULT_CANCELED 值来显示。

每当扫描模式变化的时候，应用程序可以为通过 ACTION_SCAN_MODE_CHANGED 值来监听全局的消息通知。比如，当设备停止被搜寻以后，该消息可以被系统通知给应用程序。需要 BLUETOOTH 权限。

常量值：android.bluetooth.adapter.action.REQUEST_DISCOVERABLE。

（5）String ACTION_REQUEST_ENABLE

Activity 活动：显示一个允许用户打开蓝牙模块的系统活动。当蓝牙模块完成打开工作，或者当用

户决定不打开蓝牙模块时，系统活动将返回该值。Android 运用 onActivityResult(int, int, Intent)回收方法来传递该活动结果的通知。如果蓝牙模块被打开，将通过 resultCode 值 RESULT_OK 来显示；如果用户拒绝该请求，或者设备产生了错误，则通过 RESULT_CANCELED 值来显示。每当蓝牙模块被打开或者关闭，应用程序可以为通过 ACTION_STATE_CHANGED 值来监听全局的消息通知。需要 BLUETOOTH 权限。

常量值：android.bluetooth.adapter.action.REQUEST_ENABLE。

（6）String ACTION_SCAN_MODE_CHANGED

广播活动：指明蓝牙扫描模块或者本地适配器已经发生变化。它总是包含 EXTRA_SCAN_MODE 和 EXTRA_PREVIOUS_SCAN_MODE。这两个附加域各自包含了新的和旧的扫描模式。需要 BLUETOOTH 权限。

常量值：android.bluetooth.adapter.action.SCAN_MODE_CHANGED。

（7）String ACTION_STATE_CHANGED

广播活动：本来的蓝牙适配器的状态已经改变，例如蓝牙模块已经被打开或者关闭。它总是包含 EXTRA_STATE 和 EXTRA_PREVIOUS_STATE。这两个附加域各自包含了新的和旧的状态。需要 BLUETOOTH 权限接收。

常量值：android.bluetooth.adapter.action.STATE_CHANGED。

（8）int ERROR

功能：标记该类的错误值。确保和该类中的任意其他整数常量不相等。它为需要一个标记错误值的函数提供了便利。例如：

```
Intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, BluetoothAdapter.ERROR)
```

常量值：-2147483648 (0x80000000)。

（9）String EXTRA_DISCOVERABLE_DURATION

功能：试图在 ACTION_REQUEST_DISCOVERABLE 常量中作为一个可选的整型附加域，来为短时间内的设备发现请求一个特定的持续时间。默认值为 120 秒，超过 300 秒的请求将被限制。这些值是可以变化的。

常量值：android.bluetooth.adapter.extra.DISCOVERABLE_DURATION。

（10）String EXTRA_LOCAL_NAME

功能：试图在 ACTION_LOCAL_NAME_CHANGED 常量中作为一个字符串附加域，来请求本地蓝牙的名称。

常量值：android.bluetooth.adapter.extra.LOCAL_NAME。

（11）String EXTRA_PREVIOUS_SCAN_MODE

功能：试图在 ACTION_SCAN_MODE_CHANGED 常量中作为一个整型附加域，来请求以前的扫描模式。可能值包括：

- ❑ SCAN_MODE_NONE
- ❑ SCAN_MODE_CONNECTABLE
- ❑ SCAN_MODE_CONNECTABLE_DISCOVERABLE

常量值：android.bluetooth.adapter.extra.PREVIOUS_SCAN_MODE。

（12）String EXTRA_PREVIOUS_STATE

功能：试图在 ACTION_STATE_CHANGED 常量中作为一个整型附加域，来请求以前的供电状态。

可以取得值如下。

- ❑ STATE_OFF
- ❑ STATE_TURNING_ON
- ❑ STATE_ON
- ❑ STATE_TURNING_OFF

常量值：android.bluetooth.adapter.extra.PREVIOUS_STATE。

（13）String EXTRA_SCAN_MODE

功能：试图在 ACTION_SCAN_MODE_CHANGED 常量中作为一个整型附加域，来请求当前的扫描模式，可以取得值如下。

- ❑ SCAN_MODE_NONE
- ❑ SCAN_MODE_CONNECTABLE
- ❑ SCAN_MODE_CONNECTABLE_DISCOVERABLE

常量值：android.bluetooth.adapter.extra.SCAN_MODE。

（14）String EXTRA_STATE

功能：试图在 ACTION_STATE_CHANGED 常量中作为一个整型附加域，来请求当前的供电状态。可以取得值如下。

- ❑ STATE_OFF
- ❑ STATE_TURNING_ON
- ❑ STATE_ON
- ❑ STATE_TURNING_OFF

常量值：android.bluetooth.adapter.extra.STATE。

（15）int SCAN_MODE_CONNECTABLE

功能：指明在本地蓝牙适配器中，查询扫描功能失效，但页面扫描功能有效。因此该设备不能被远程蓝牙设备发现，但如果以前曾经发现过该设备，则远程设备可以对其进行连接。

常量值：21 (0x00000015)。

（16）int SCAN_MODE_CONNECTABLE_DISCOVERABLE

功能：指明在本地蓝牙适配器中，查询扫描功能和页面扫描功能都有效。因此该设备既可以被远程蓝牙设备发现，也可以被其连接。

常量值：23 (0x00000017)。

（17）int SCAN_MODE_NONE

功能：指明在本地蓝牙适配器中，查询扫描功能和页面扫描功能都失效。因此该设备既不可以被远程蓝牙设备发现，也不可以被其连接。

常量值：20 (0x00000014)。

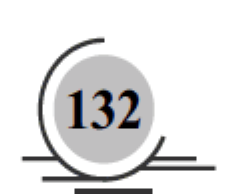
（18）int STATE_OFF

功能：指明本地蓝牙适配器模块已经关闭。

常量值：10 (0x0000000a)。

（19）int STATE_ON

功能：指明本地蓝牙适配器模块已经打开，并且准备被使用。



（20）int STATE_TURNING_OFF

功能：指明本地蓝牙适配器模块正在关闭。本地客户端可以立刻尝试友好地断开任意外部连接。

常量值：13 (0x0000000d)。

（21）int STATE_TURNING_ON

功能：指明本地蓝牙适配器模块正在打开，然而本地客户在尝试使用这个适配器之前需要为STATE_ON 状态而等待。

常量值：11 (0x0000000b)。

3. BluetoothAdapter 类的公共方法

（1）public boolean cancelDiscovery ()

功能：取消当前的设备发现查找进程需要 BLUETOOTH_ADMIN 权限。因为对蓝牙适配器而言，查找是一个重量级的过程，因此这个方法必须在尝试连接到远程设备前使用 connect()方法进行调用。发现的过程不会由活动来进行管理，但是它会作为一个系统服务来运行，因此即使它不能直接请求这样的一个查询动作，也必须取消该搜索进程。如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON。

返回值：成功则返回 true，错误则返回 false。

（2）public static boolean checkBluetoothAddress (String address)

功能：验证皆如"00:43:A8:23:10:F0"之类的蓝牙地址，字母必须为大写才有效。

参数 address：字符串形式的蓝牙模块地址。

返回值：地址正确则返回 true，否则返回 false。

（3）public boolean disable ()

功能：关闭本地蓝牙适配器——不能在没有明确关闭蓝牙的用户动作中使用。这个方法友好地停止所有的蓝牙连接，停止蓝牙系统服务，以及对所有基础蓝牙硬件进行断电。没有用户的直接同意，蓝牙不能永远被禁止。这个 disable()方法只提供了一个应用，该应用包含了一个改变系统设置的用户界面（例如“电源控制”应用）。

这是一个异步调用方法：该方法将马上获得返回值，用户要通过监听 ACTION_STATE_CHANGED 值来获取随后的适配器状态改变的通知。如果该调用返回 true 值，则该适配器状态会立刻从 STATE_ON 转向 STATE_TURNING_OFF，稍后则会转为 STATE_OFF 或者 STATE_ON。如果该调用返回 false，那么系统已经有一个保护蓝牙适配器被关闭的问题，比如该适配器已经被关闭了。

需要 BLUETOOTH_ADMIN 权限。

返回值：如果蓝牙适配器的停止进程已经开启则返回 true，如果产生错误则返回 false。

（4）public boolean enable ()

功能：打开本地蓝牙适配器——不能在没有明确打开蓝牙的用户动作中使用。该方法将为基础的蓝牙硬件供电，并且启动所有的蓝牙系统服务。没有用户的直接同意，蓝牙不能永远被禁止。如果用户为了创建无线连接而打开了蓝牙模块，则其需要 ACTION_REQUEST_ENABLE 值，该值将提出一个请求用户允许以打开蓝牙模块的会话。这个 enable()值只提供了一个应用，该应用包含了一个改变系统设置的用户界面（如“电源控制”应用）。

这是一个异步调用方法：该方法将马上获得返回值，用户要通过监听 ACTION_STATE_CHANGED 值来获取随后的适配器状态改变的通知。如果该调用返回 true 值，则该适配器状态会立刻从 STATE_OFF

转向 STATE_TURNING_ON，稍后则会转为 STATE_OFF 或者 STATE_ON。如果该调用返回 false，那么说明系统已经有一个保护蓝牙适配器被打开的问题，比如飞行模式，或者该适配器已经被打开。

需要 BLUETOOTH_ADMIN 权限。

返回值：如果蓝牙适配器的打开进程已经开启则返回 true，如果产生错误则返回 false。

（5）public String getAddress ()

功能：返回本地蓝牙适配器的硬件地址。例如：

```
00:11:22:AA:BB:CC
```

需要 BLUETOOTH 权限。

返回值：字符串形式的蓝牙模块地址。

（6）public Set<BluetoothDevice> getBondedDevices ()

功能：返回已经匹配到本地适配器的 BluetoothDevice 类的对象集合。如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON。需要 BLUETOOTH 权限。

返回值：未被修改的 BluetoothDevice 类的对象集合，如果有错误则返回 null。

（7）public static synchronized BluetoothAdapter getDefaultAdapter ()

功能：获取对默认本地蓝牙适配器的操作权限。目前 Android 只支持一个蓝牙适配器，但是 API 可以被扩展为支持多个适配器。该方法总是返回默认的适配器。

返回值：返回默认的本地适配器，如果蓝牙适配器在该硬件平台上不能被支持，则返回 null。

（8）public String getName ()

功能：获取本地蓝牙适配器的蓝牙名称，这个名称对于外界蓝牙设备而言是可见的。需要 BLUETOOTH 权限。

返回值：该蓝牙适配器名称，如果有错误则返回 null。

（9）public BluetoothDevice getRemoteDevice (String address)

功能：为给予的蓝牙硬件地址获取一个 BluetoothDevice 对象。合法的蓝牙硬件地址必须为大写，格式类似于"00:11:22:33:AA:BB"。checkBluetoothAddress(String)方法可以用来验证蓝牙地址的正确性。BluetoothDevice 类对于合法的硬件地址总会产生返回值，即使这个适配器从未见过该设备。

参数：address 合法的蓝牙 MAC 地址。

异常：IllegalArgumentException，如果地址不合法。

（10）public int getScanMode ()

功能：获取本地蓝牙适配器的当前蓝牙扫描模式，蓝牙扫描模式决定本地适配器可连接并且/或者可被远程蓝牙设备所连接。需要 BLUETOOTH 权限，可能的取值如下。

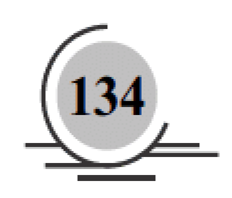
- ❑ SCAN_MODE_NONE
- ❑ SCAN_MODE_CONNECTABLE
- ❑ SCAN_MODE_CONNECTABLE_DISCOVERABLE

如果蓝牙状态不是 STATE_ON，则这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON。

返回值：扫描模式。

（11）public int getState ()

功能：获取本地蓝牙适配器的当前状态，需要 BLUETOOTH 类。可能的取值如下。



- STATE_OFF
- STATE_TURNING_ON
- STATE_ON
- STATE_TURNING_OFF

返回值：蓝牙适配器的当前状态。

（12）public boolean isDiscovering ()

功能：如果当前蓝牙适配器正处于设备发现查找进程中，则返回真值。设备查找是一个重量级过程。当查找正在进行的时候，用户不能尝试对新的远程蓝牙设备进行连接，同时存在的连接将获得有限制的带宽以及高等待时间。用户可用 cencelDiscovery()方法来取消正在执行的查找进程。

应用程序也可以为 ACTION_DISCOVERY_STARTED 或者 ACTION_DISCOVERY_FINISHED 进行注册，从而当查找开始或者完成的时候，可以获得通知。

如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON。需要 BLUETOOTH 权限。

返回值：如果正在查找，则返回 true。

（13）public boolean isEnabled ()

功能：如果蓝牙正处于打开状态并可用，则返回真值，与 getBluetoothState()==STATE_ON 等价需要 BLUETOOTH 权限。

返回值：如果本地适配器已经打开，则返回 true。

（14）public BluetoothServerSocket listenUsingRfcommWithServiceRecord (String name, UUID uuid)

功能：创建一个正在监听的、安全的、带有服务记录的无线射频通信（RFCOMM）蓝牙端口。一个对该端口进行连接的远程设备将被认证，对该端口的通信将被加密。使用 accpet()方法可以获取从监听 BluetoothServerSocket 处新来的连接。该系统分配一个未被使用的无线射频通信通道来进行监听。

该系统也将注册一个服务探索协议（SDP）记录，该记录带有一个包含了特定的通用唯一识别码（Universally Unique Identifier，UUID），服务器名称和自动分配通道的本地 SDP 服务。远程蓝牙设备可以用相同的 UUID 来查询自己的 SDP 服务器，并搜寻连接到了哪个通道上。如果该端口已经关闭，或者该应用程序异常退出，则这个 SDP 记录会被移除。使用 createRfcommSocketTo ServiceRecord(UUID)可以从另一使用相同 UUID 的设备来连接到这个端口，需要 BLUETOOTH 权限。

参数说明如下。

- name：SDP记录下的服务器名。
- uuid：SDP记录下的UUID。

返回值：一个正在监听的无线射频通信蓝牙服务端口。

异常：IOException，表示产生错误，比如蓝牙设备不可用，或者许可无效，或者通道被占用。

（15）public boolean setName (String name)

功能：设置蓝牙或者本地蓝牙适配器的昵称，这个名字对于外界蓝牙设备而言是可见的。合法的蓝牙名称最多拥有 248 位 UTF-8 字符，但是很多外界设备只能显示前 40 个字符，有些可能只限制前 20 个字符。

如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON。需要 BLUETOOTH_ADMIN 权限。

参数 name：一个合法的蓝牙名称。

返回值：如果该名称已被设定，则返回 true，否则返回 false。

（16）public boolean startDiscovery ()

功能：开始对远程设备进行查找的进程，它通常涉及一个大概需时 12 秒的查询扫描过程，紧跟着是一个对每个获取到自身蓝牙名称的新设备的页面扫描。这是一个异步调用方法：该方法将马上获得返回值，注册 ACTION_DISCOVERY_STARTED and ACTION_DISCOVERY_FINISHED 意图准确地确定该探索是处于开始阶段或者完成阶段。注册 ACTION_FOUND 以活动远程蓝牙设备已找到的通知。

设备查找是一个重量级过程。当查找正在进行的时候，用户不能尝试对新的远程蓝牙设备进行连接，同时存在的连接将获得有限制的带宽以及高等待时间。用户可用 cencelDiscovery()方法来取消正在执行的查找进程。发现的过程不会由活动来进行管理，但是它会作为一个系统服务来运行，因此即使它不能直接请求这样的一个查询动作，也必须取消该搜索进程。设备搜寻只寻找已经被连接的远程设备。许多蓝牙设备默认不会被搜寻到，并且需要进入到一个特殊的模式当中。

如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON。需要 BLUETOOTH_ADMIN 权限。

返回值：成功返回 true，错误返回 false。

6.4.4 BluetoothClass.Service 类

BluetoothClass.Service 类的格式如下所示。

```
public static final class BluetoothClass.Service extends Object
```

BluetoothClass.Service 类的结构如下所示。

```
java.lang.Object
android.bluetooth.BluetoothClass.Service
```

BluetoothClass.Service 类用于定义所有的服务类常量，任意 BluetoothClass 由 0 或多个服务类编码组成。在 BluetoothClass.Service 类中包含如下常量。

- int AUDIO
- int CAPTURE
- int INFORMATION
- int LIMITED_DISCOVERABILITY
- int NETWORKING
- int OBJECT_TRANSFER
- int POSITIONING
- int RENDER
- int TELEPHONY

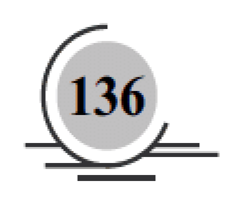
6.4.5 BluetoothClass.Device 类

BluetoothClass.Device 类的格式如下所示。

```
public final class BluetoothClass.Device extends Object
```

BluetoothClass.Device 类的结构如下所示。

```
java.lang.Object
android.bluetooth.BluetoothClass.Device
```



BluetoothClass.Device 类用于定义所有的设备类的常量，每个 BluetoothClass 有一个带有主要和较小部分的设备类进行编码。里面的常量代表主要和较小的设备类部分（完整的设备类）的组合。BluetoothClass.Device.Major 的常量只能代表主要设备类。

BluetoothClass.Device 有一个内部类，此内部类定义了所有的主要设备类常量。内部类的定义格式如下所示。

```
class BluetoothClass.Device.Major
```

注意： 到此为止，Android中的蓝牙类介绍完毕。我们在调用这些类时，除了首先确保API Level至少为版本5以上，并且还需注意添加相应的权限，比如在使用通信时，需要在文件androidmanifest.xml中加入<uses-permission android:name="android.permission.BLUETOOTH" />权限，而在开关蓝牙时需要类似android.permission.BLUETOOTH_ADMIN权限。

6.5　Android BlueDroid 架构详解

知识点讲解：光盘:视频\知识点\第 6 章\Android BlueDroid 架构详解.avi

了解了 Android 系统中低功耗蓝牙协议栈 BlueDroid 的基本知识后，在本节的内容中，将详细讲解 Android 源码中低功耗协议栈 BlueDroid 的具体架构知识，为读者学习本书后面的知识打下基础。

6.5.1　Android 系统中 BlueDroid 的架构

在 Android 新系统中，采用 BlueDroid 作为默认的协议栈，BlueDroid 分为如下两个部分。

- Bluetooth Embedded System（BTE）：实现了BT的核心功能。
- Bluetooth Application Layer（BTA）：用于和Android Framework层进行交互。

在 Android 新系统中，BT 系统服务通过 JNI 与 BT stack 进行交互，并且通过 Binder IPC 通信与应用交互，这个系统服务同时也提供给 RD 获取不同的 BT profiles。如图 6-5 展示了 BT stack 的一个大体的结构。

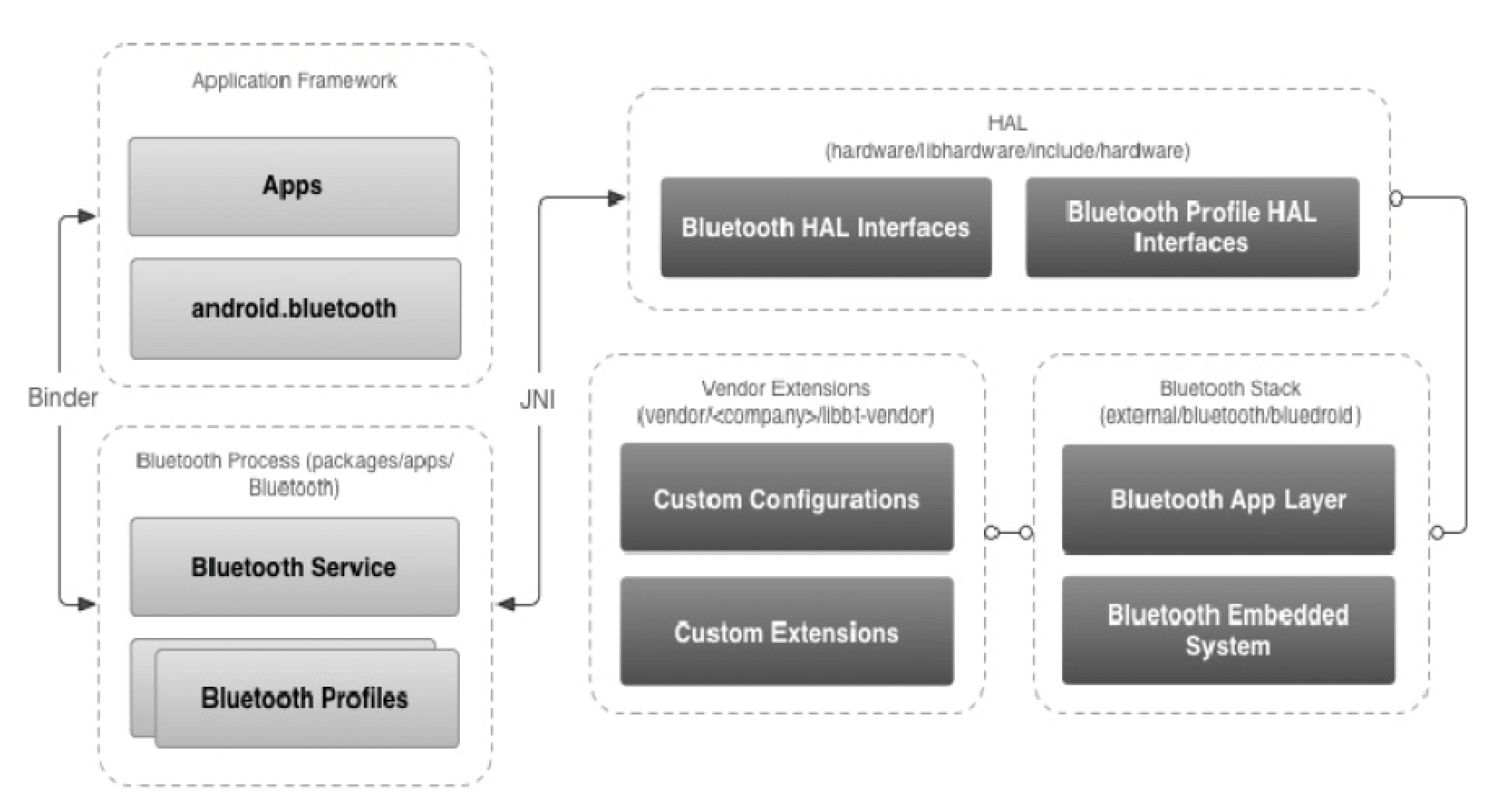


图 6-5　BT stack 的结构

6.5.2 Application Framework 层分析

在 Application Framework 层中，功能是利用 android.bluetooth APIS 和 Bluetooth Hardware 层进行交互，也就是通过 Binder IPC 机制调用 Bluetooth 进程。Application Framework 层的代码位于如下所示的目录中。

```
framework/base/core/java/android.bluetooth/
```

在文件 framework/base/core/java/android/bluetooth/BluetoothA2dp.java 中定义了 connect(BluetoothDevice device)方法，功能是通过 Binder IPC 通信机制调用如下文件中的一个内部私有类。

```
packages/apps/Bluetooth/src/com/android/bluetooth/a2dp/A2dpService.java
```

文件 BluetoothA2dp.java 的具体实现代码如下所示。

```
public final class BluetoothA2dp implements BluetoothProfile {
    private static final String TAG = "BluetoothA2dp";
    private static final boolean DBG = true;
    private static final boolean VDBG = false;
    @SdkConstant(SdkConstantType.BROADCAST_INTENT_ACTION)
    public static final String ACTION_CONNECTION_STATE_CHANGED =
        "android.bluetooth.a2dp.profile.action.CONNECTION_STATE_CHANGED";
    @SdkConstant(SdkConstantType.BROADCAST_INTENT_ACTION)
    public static final String ACTION_PLAYING_STATE_CHANGED =
        "android.bluetooth.a2dp.profile.action.PLAYING_STATE_CHANGED";
    public static final int STATE_PLAYING     =   10;
    public static final int STATE_NOT_PLAYING     =   11;

    private Context mContext;
    private ServiceListener mServiceListener;
    private IBluetoothA2dp mService;
    private BluetoothAdapter mAdapter;

    final private IBluetoothStateChangeCallback mBluetoothStateChangeCallback =
            new IBluetoothStateChangeCallback.Stub() {
                public void onBluetoothStateChange(boolean up) {
                    if (DBG) Log.d(TAG, "onBluetoothStateChange: up=" + up);
                    if (!up) {
                        if (VDBG) Log.d(TAG,"Unbinding service...");
                        synchronized (mConnection) {
                            try {
                                mService = null;
                                mContext.unbindService(mConnection);
                            } catch (Exception re) {
                                Log.e(TAG,"",re);
                            }
                        }
                    } else {
                        synchronized (mConnection) {
                            try {
                                if (mService == null) {
                                    if (VDBG) Log.d(TAG,"Binding service...");
```

```
                                    if (!mContext.bindService(new Intent(IBluetoothA2dp.class.getName()),
mConnection, 0)) {
                                        Log.e(TAG, "Could not bind to Bluetooth A2DP Service");
                                    }
                                }
                            } catch (Exception re) {
                                Log.e(TAG,"",re);
                            }
                        }
                    }
                }
        };
BluetoothA2dp(Context context, ServiceListener l) {
        mContext = context;
        mServiceListener = l;
        mAdapter = BluetoothAdapter.getDefaultAdapter();
        IBluetoothManager mgr = mAdapter.getBluetoothManager();
        if (mgr != null) {
            try {
                mgr.registerStateChangeCallback(mBluetoothStateChangeCallback);
            } catch (RemoteException e) {
                Log.e(TAG,"",e);
            }
        }

        if (!context.bindService(new Intent(IBluetoothA2dp.class.getName()), mConnection, 0)) {
            Log.e(TAG, "Could not bind to Bluetooth A2DP Service");
        }
    }

void close() {
        mServiceListener = null;
        IBluetoothManager mgr = mAdapter.getBluetoothManager();
        if (mgr != null) {
            try {
                mgr.unregisterStateChangeCallback(mBluetoothStateChangeCallback);
            } catch (Exception e) {
                Log.e(TAG,"",e);
            }
        }

        synchronized (mConnection) {
            if (mService != null) {
                try {
                    mService = null;
                    mContext.unbindService(mConnection);
                } catch (Exception re) {
                    Log.e(TAG,"",re);
                }
            }
        }
```

```
}

public void finalize() {
    close();
}
public boolean connect(BluetoothDevice device) {
    if (DBG) log("connect(" + device + ")");
    if (mService != null && isEnabled() &&
        isValidDevice(device)) {
        try {
            return mService.connect(device);
        } catch (RemoteException e) {
            Log.e(TAG, "Stack:" + Log.getStackTraceString(new Throwable()));
            return false;
        }
    }
    if (mService == null) Log.w(TAG, "Proxy not attached to service");
    return false;
}
public boolean disconnect(BluetoothDevice device) {
    if (DBG) log("disconnect(" + device + ")");
    if (mService != null && isEnabled() &&
        isValidDevice(device)) {
        try {
            return mService.disconnect(device);
        } catch (RemoteException e) {
            Log.e(TAG, "Stack:" + Log.getStackTraceString(new Throwable()));
            return false;
        }
    }
    if (mService == null) Log.w(TAG, "Proxy not attached to service");
    return false;
}

/**
 * {@inheritDoc}
 */
public List<BluetoothDevice> getConnectedDevices() {
    if (VDBG) log("getConnectedDevices()");
    if (mService != null && isEnabled()) {
        try {
            return mService.getConnectedDevices();
        } catch (RemoteException e) {
            Log.e(TAG, "Stack:" + Log.getStackTraceString(new Throwable()));
            return new ArrayList<BluetoothDevice>();
        }
    }
    if (mService == null) Log.w(TAG, "Proxy not attached to service");
    return new ArrayList<BluetoothDevice>();
}
public List<BluetoothDevice> getDevicesMatchingConnectionStates(int[ ] states) {
```

```
        if (VDBG) log("getDevicesMatchingStates()");
        if (mService != null && isEnabled()) {
            try {
                return mService.getDevicesMatchingConnectionStates(states);
            } catch (RemoteException e) {
                Log.e(TAG, "Stack:" + Log.getStackTraceString(new Throwable()));
                return new ArrayList<BluetoothDevice>();
            }
        }
        if (mService == null) Log.w(TAG, "Proxy not attached to service");
        return new ArrayList<BluetoothDevice>();
    }
    public int getConnectionState(BluetoothDevice device) {
        if (VDBG) log("getState(" + device + ")");
        if (mService != null && isEnabled()
            && isValidDevice(device)) {
            try {
                return mService.getConnectionState(device);
            } catch (RemoteException e) {
                Log.e(TAG, "Stack:" + Log.getStackTraceString(new Throwable()));
                return BluetoothProfile.STATE_DISCONNECTED;
            }
        }
        if (mService == null) Log.w(TAG, "Proxy not attached to service");
        return BluetoothProfile.STATE_DISCONNECTED;
    }
    public boolean setPriority(BluetoothDevice device, int priority) {
        if (DBG) log("setPriority(" + device + ", " + priority + ")");
        if (mService != null && isEnabled()
            && isValidDevice(device)) {
            if (priority != BluetoothProfile.PRIORITY_OFF &&
                priority != BluetoothProfile.PRIORITY_ON){
              return false;
            }
            try {
                return mService.setPriority(device, priority);
            } catch (RemoteException e) {
                Log.e(TAG, "Stack:" + Log.getStackTraceString(new Throwable()));
                return false;
            }
        }
        if (mService == null) Log.w(TAG, "Proxy not attached to service");
        return false;
    }
    public int getPriority(BluetoothDevice device) {
        if (VDBG) log("getPriority(" + device + ")");
        if (mService != null && isEnabled()
            && isValidDevice(device)) {
            try {
                return mService.getPriority(device);
            } catch (RemoteException e) {
```

```
                Log.e(TAG, "Stack:" + Log.getStackTraceString(new Throwable()));
                return BluetoothProfile.PRIORITY_OFF;
            }
        }
        if (mService == null) Log.w(TAG, "Proxy not attached to service");
        return BluetoothProfile.PRIORITY_OFF;
    }
    public boolean isA2dpPlaying(BluetoothDevice device) {
        if (mService != null && isEnabled()
            && isValidDevice(device)) {
            try {
                return mService.isA2dpPlaying(device);
            } catch (RemoteException e) {
                Log.e(TAG, "Stack:" + Log.getStackTraceString(new Throwable()));
                return false;
            }
        }
        if (mService == null) Log.w(TAG, "Proxy not attached to service");
        return false;
    }
    public boolean shouldSendVolumeKeys(BluetoothDevice device) {
        if (isEnabled() && isValidDevice(device)) {
            ParcelUuid[ ] uuids = device.getUuids();
            if (uuids == null) return false;

            for (ParcelUuid uuid: uuids) {
                if (BluetoothUuid.isAvrcpTarget(uuid)) {
                    return true;
                }
            }
        }
        return false;
    }
    public static String stateToString(int state) {
        switch (state) {
        case STATE_DISCONNECTED:
            return "disconnected";
        case STATE_CONNECTING:
            return "connecting";
        case STATE_CONNECTED:
            return "connected";
        case STATE_DISCONNECTING:
            return "disconnecting";
        case STATE_PLAYING:
            return "playing";
        case STATE_NOT_PLAYING:
          return "not playing";
        default:
            return "<unknown state " + state + ">";
        }
    }
```

```
    private ServiceConnection mConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className, IBinder service) {
            if (DBG) Log.d(TAG, "Proxy object connected");
            mService = IBluetoothA2dp.Stub.asInterface(service);

            if (mServiceListener != null) {
                mServiceListener.onServiceConnected(BluetoothProfile.A2DP, BluetoothA2dp.this);
            }
        }
        public void onServiceDisconnected(ComponentName className) {
            if (DBG) Log.d(TAG, "Proxy object disconnected");
            mService = null;
            if (mServiceListener != null) {
                mServiceListener.onServiceDisconnected(BluetoothProfile.A2DP);
            }
        }
    };
```

在上述代码中，定义了 A2dpService 对象 service，并调用 getService()方法。A2dpService 是一个继承于类 ProfileService 的子类，而 ProfileService 是继承于类 Service 的子类。文件 A2dpService.java 的主要实现代码如下所示。

```
public class A2dpService extends ProfileService {
    private static final boolean DBG = false;
    private static final String TAG="A2dpService";

    private A2dpStateMachine mStateMachine;
    private Avrcp mAvrcp;
    private static A2dpService sAd2dpService;

    protected String getName() {
        return TAG;
    }

    protected IProfileServiceBinder initBinder() {
        return new BluetoothA2dpBinder(this);
    }

    protected boolean start() {
        mStateMachine = A2dpStateMachine.make(this, this);
        mAvrcp = Avrcp.make(this);
        setA2dpService(this);
        return true;
    }

    protected boolean stop() {
        mStateMachine.doQuit();
        mAvrcp.doQuit();
        return true;
    }
```

```
protected boolean cleanup() {
    if (mStateMachine!= null) {
        mStateMachine.cleanup();
    }
    if (mAvrcp != null) {
        mAvrcp.cleanup();
        mAvrcp = null;
    }
    clearA2dpService();
    return true;
}

//API Methods

public static synchronized A2dpService getA2dpService(){
    if (sAd2dpService != null && sAd2dpService.isAvailable()) {
        if (DBG) Log.d(TAG, "getA2DPService(): returning " + sAd2dpService);
        return sAd2dpService;
    }
    if (DBG)   {
        if (sAd2dpService == null) {
            Log.d(TAG, "getA2dpService(): service is NULL");
        } else if (!(sAd2dpService.isAvailable())) {
            Log.d(TAG,"getA2dpService(): service is not available");
        }
    }
    return null;
}

private static synchronized void setA2dpService(A2dpService instance) {
    if (instance != null && instance.isAvailable()) {
        if (DBG) Log.d(TAG, "setA2dpService(): set to: " + sAd2dpService);
        sAd2dpService = instance;
    } else {
        if (DBG)   {
            if (sAd2dpService == null) {
                Log.d(TAG, "setA2dpService(): service not available");
            } else if (!sAd2dpService.isAvailable()) {
                Log.d(TAG,"setA2dpService(): service is cleaning up");
            }
        }
    }
}

private static synchronized void clearA2dpService() {
    sAd2dpService = null;
}

public boolean connect(BluetoothDevice device) {
    enforceCallingOrSelfPermission(BLUETOOTH_ADMIN_PERM,
                                   "Need BLUETOOTH ADMIN permission");
```

```
            if (getPriority(device) == BluetoothProfile.PRIORITY_OFF) {
                return false;
            }

            int connectionState = mStateMachine.getConnectionState(device);
            if (connectionState == BluetoothProfile.STATE_CONNECTED ||
                connectionState == BluetoothProfile.STATE_CONNECTING) {
                return false;
            }

            mStateMachine.sendMessage(A2dpStateMachine.CONNECT, device);
            return true;
        }

        boolean disconnect(BluetoothDevice device) {
            enforceCallingOrSelfPermission(BLUETOOTH_ADMIN_PERM,
                                           "Need BLUETOOTH ADMIN permission");
            int connectionState = mStateMachine.getConnectionState(device);
            if (connectionState != BluetoothProfile.STATE_CONNECTED &&
                connectionState != BluetoothProfile.STATE_CONNECTING) {
                return false;
            }

            mStateMachine.sendMessage(A2dpStateMachine.DISCONNECT, device);
            return true;
        }
```

由此可见，在接下来的通信过程中，通过 Binder IPC 通信机制调用了文件 A2dpService.java 中的 connect (BluetoothDevice device)方法。上述过程就是 Bluetooth Application Framework 与 Bluetooth Process 之间的调用过程。

6.5.3　分析 Bluetooth System Service 层

在 Android 系统中，Bluetooth System Service 位于 packages/apps/Bluetooth 目录下，将其打包成一个 Android App（Android 应用程序）包，并且在 Android Framework 层实现 BT Service 和各种 profile。BT App 接下来会通过 JNI 调用到 HAL 层。

在文件 A2dpService.java 中，connect 方法会发送一个 StateMachine.sendMessage(A2dpStateMachine.CONNECT, device)的 message（信息 0，这个 message 会被 A2dpStateMachine 对象的 processMessage(Message)方法接收到），对应代码如下所示。

```
case CONNECT:
                BluetoothDevice device = (BluetoothDevice) message.obj;
                broadcastConnectionState(device, BluetoothProfile.STATE_CONNECTING,
                                BluetoothProfile.STATE_DISCONNECTED);

                if (!connectA2dpNative(getByteAddress(device)) ) {
                    broadcastConnectionState(device, BluetoothProfile.STATE_DISCONNECTED,
                                    BluetoothProfile.STATE_CONNECTING);
```

```
                break;
            }

            synchronized (A2dpStateMachine.this) {
                mTargetDevice = device;
                transitionTo(mPending);
            }
            // TODO(BT) remove CONNECT_TIMEOUT when the stack
            //            sends back events consistently
            sendMessageDelayed(CONNECT_TIMEOUT, 30000);
            break;
```

在上述代码中，会通过“connectA2dpNative(getByteAddress(device));”代码行设置通过 JNI 调用到 Native（本地程序）：

```
private native boolean connectA2dpNative(byte[ ] address);
```

6.5.4 JNI 层详解

在 Android 系统中，和 Bluetooth 有关的 JNI 代码位于如下所示的目录中。

```
packages/apps/bluetooth/jni
```

JNI 层的代码会调用到 HAL 层，并且在确信一些 BT 操作被触发时从 HAL 获取一些回调，例如当 BT 设备被发现时。例如在 A2dp 连接的例子中，BT System Service 会通过 JNI 调用文件 com_android_bluetooth_a2dp.cpp 中的方法，此文件的主要实现代码如下所示。

```
namespace android {
static jmethodID method_onConnectionStateChanged;
static jmethodID method_onAudioStateChanged;

static const btav_interface_t *sBluetoothA2dpInterface = NULL;
static jobject mCallbacksObj = NULL;
static JNIEnv *sCallbackEnv = NULL;

static bool checkCallbackThread() {
    // Always fetch the latest callbackEnv from AdapterService.
    // Caching this could cause this sCallbackEnv to go out-of-sync
    // with the AdapterService's ENV if an ASSOCIATE/DISASSOCIATE event
    // is received
    //if (sCallbackEnv == NULL) {
    sCallbackEnv = getCallbackEnv();
    // }

    JNIEnv* env = AndroidRuntime::getJNIEnv();
    if (sCallbackEnv != env || sCallbackEnv == NULL) return false;
    return true;
}

static void bta2dp_connection_state_callback(btav_connection_state_t state, bt_bdaddr_t* bd_addr) {
    jbyteArray addr;

    ALOGI("%s", __FUNCTION__);
```

```
    if (!checkCallbackThread()) {                                              \
        ALOGE("Callback: '%s' is not called on the correct thread", __FUNCTION__);    \
        return;                                                                \
    }
    addr = sCallbackEnv->NewByteArray(sizeof(bt_bdaddr_t));
    if (!addr) {
        ALOGE("Fail to new jbyteArray bd addr for connection state");
        checkAndClearExceptionFromCallback(sCallbackEnv, __FUNCTION__);
        return;
    }

    sCallbackEnv->SetByteArrayRegion(addr, 0, sizeof(bt_bdaddr_t), (jbyte*) bd_addr);
    sCallbackEnv->CallVoidMethod(mCallbacksObj, method_onConnectionStateChanged, (jint) state,
                                 addr);
    checkAndClearExceptionFromCallback(sCallbackEnv, __FUNCTION__);
    sCallbackEnv->DeleteLocalRef(addr);
}

static void bta2dp_audio_state_callback(btav_audio_state_t state, bt_bdaddr_t* bd_addr) {
    jbyteArray addr;

    ALOGI("%s", __FUNCTION__);

    if (!checkCallbackThread()) {                                              \
        ALOGE("Callback: '%s' is not called on the correct thread", __FUNCTION__);    \
        return;                                                                \
    }
    addr = sCallbackEnv->NewByteArray(sizeof(bt_bdaddr_t));
    if (!addr) {
        ALOGE("Fail to new jbyteArray bd addr for connection state");
        checkAndClearExceptionFromCallback(sCallbackEnv, __FUNCTION__);
        return;
    }

    sCallbackEnv->SetByteArrayRegion(addr, 0, sizeof(bt_bdaddr_t), (jbyte*) bd_addr);
    sCallbackEnv->CallVoidMethod(mCallbacksObj, method_onAudioStateChanged, (jint) state,
                                 addr);
    checkAndClearExceptionFromCallback(sCallbackEnv, __FUNCTION__);
    sCallbackEnv->DeleteLocalRef(addr);
}

static btav_callbacks_t sBluetoothA2dpCallbacks = {
    sizeof(sBluetoothA2dpCallbacks),
    bta2dp_connection_state_callback,
    bta2dp_audio_state_callback
};

static void classInitNative(JNIEnv* env, jclass clazz) {
    int err;
    const bt_interface_t* btInf;
```

```
    bt_status_t status;

    method_onConnectionStateChanged =
        env->GetMethodID(clazz, "onConnectionStateChanged", "(I[B)V");

    method_onAudioStateChanged =
        env->GetMethodID(clazz, "onAudioStateChanged", "(I[B)V");
    /*
    if ( (btInf = getBluetoothInterface()) == NULL) {
        ALOGE("Bluetooth module is not loaded");
        return;
    }

    if ( (sBluetoothA2dpInterface = (btav_interface_t *)
            btInf->get_profile_interface(BT_PROFILE_ADVANCED_AUDIO_ID)) == NULL) {
        ALOGE("Failed to get Bluetooth A2DP Interface");
        return;
    }
    */

    // TODO(BT) do this only once or
    //          Do we need to do this every time the BT reenables?
    /*
    if ( (status = sBluetoothA2dpInterface->init(&sBluetoothA2dpCallbacks)) != BT_STATUS_SUCCESS) {
        ALOGE("Failed to initialize Bluetooth A2DP, status: %d", status);
        sBluetoothA2dpInterface = NULL;
        return;
    }*/

    ALOGI("%s: succeeds", __FUNCTION__);
}

static void initNative(JNIEnv *env, jobject object) {
    const bt_interface_t* btInf;
    bt_status_t status;

    if ( (btInf = getBluetoothInterface()) == NULL) {
        ALOGE("Bluetooth module is not loaded");
        return;
    }

    if (sBluetoothA2dpInterface !=NULL) {
         ALOGW("Cleaning up A2DP Interface before initializing...");
         sBluetoothA2dpInterface->cleanup();
         sBluetoothA2dpInterface = NULL;
    }

    if (mCallbacksObj != NULL) {
         ALOGW("Cleaning up A2DP callback object");
         env->DeleteGlobalRef(mCallbacksObj);
         mCallbacksObj = NULL;
```

```
    }

    if ( (sBluetoothA2dpInterface = (btav_interface_t *)
          btInf->get_profile_interface(BT_PROFILE_ADVANCED_AUDIO_ID)) == NULL) {
        ALOGE("Failed to get Bluetooth A2DP Interface");
        return;
    }

    if ( (status = sBluetoothA2dpInterface->init(&sBluetoothA2dpCallbacks)) != BT_STATUS_SUCCESS) {
        ALOGE("Failed to initialize Bluetooth A2DP, status: %d", status);
        sBluetoothA2dpInterface = NULL;
        return;
    }

    mCallbacksObj = env->NewGlobalRef(object);
}

static void cleanupNative(JNIEnv *env, jobject object) {
    const bt_interface_t* btInf;
    bt_status_t status;

    if ( (btInf = getBluetoothInterface()) == NULL) {
        ALOGE("Bluetooth module is not loaded");
        return;
    }

    if (sBluetoothA2dpInterface !=NULL) {
        sBluetoothA2dpInterface->cleanup();
        sBluetoothA2dpInterface = NULL;
    }

    if (mCallbacksObj != NULL) {
        env->DeleteGlobalRef(mCallbacksObj);
        mCallbacksObj = NULL;
    }
}

static jboolean connectA2dpNative(JNIEnv *env, jobject object, jbyteArray address) {
    jbyte *addr;
    bt_bdaddr_t * btAddr;
    bt_status_t status;

    ALOGI("%s: sBluetoothA2dpInterface: %p", __FUNCTION__, sBluetoothA2dpInterface);
    if (!sBluetoothA2dpInterface) return JNI_FALSE;

    addr = env->GetByteArrayElements(address, NULL);
    btAddr = (bt_bdaddr_t *) addr;
    if (!addr) {
        jniThrowIOException(env, EINVAL);
        return JNI_FALSE;
    }
```

```
    if ((status = sBluetoothA2dpInterface->connect((bt_bdaddr_t *)addr)) != BT_STATUS_SUCCESS) {
        ALOGE("Failed HF connection, status: %d", status);
    }
    env->ReleaseByteArrayElements(address, addr, 0);
    return (status == BT_STATUS_SUCCESS) ? JNI_TRUE : JNI_FALSE;
}

static jboolean disconnectA2dpNative(JNIEnv *env, jobject object, jbyteArray address) {
    jbyte *addr;
    bt_status_t status;

    if (!sBluetoothA2dpInterface) return JNI_FALSE;

    addr = env->GetByteArrayElements(address, NULL);
    if (!addr) {
        jniThrowIOException(env, EINVAL);
        return JNI_FALSE;
    }

    if ( (status = sBluetoothA2dpInterface->disconnect((bt_bdaddr_t *)addr)) != BT_STATUS_SUCCESS) {
        ALOGE("Failed HF disconnection, status: %d", status);
    }
    env->ReleaseByteArrayElements(address, addr, 0);
    return (status == BT_STATUS_SUCCESS) ? JNI_TRUE : JNI_FALSE;
}

static JNINativeMethod sMethods[ ] = {
    {"classInitNative", "()V", (void *) classInitNative},
    {"initNative", "()V", (void *) initNative},
    {"cleanupNative", "()V", (void *) cleanupNative},
    {"connectA2dpNative", "([B)Z", (void *) connectA2dpNative},
    {"disconnectA2dpNative", "([B)Z", (void *) disconnectA2dpNative},
};

int register_com_android_bluetooth_a2dp(JNIEnv* env)
{
    return jniRegisterNativeMethods(env, "com/android/bluetooth/a2dp/A2dpStateMachine",
                                    sMethods, NELEM(sMethods));
}

}
```

在上述代码中用到了结构体对象 sBluetoothA2dpInterface，此对象在方法 initNative(JNIEnv *env, jobject object)中定义获取，即如下所示的代码。

```
if ( (sBluetoothA2dpInterface = (btav_interface_t *)
      btInf->get_profile_interface(BT_PROFILE_ADVANCED_AUDIO_ID)) == NULL) {
    ALOGE("Failed to get Bluetooth A2DP Interface");
    return;
}
```

6.5.5　HAL 层详解

硬件抽象层用于定义 android.bluetooth APIs 和 BT process 调用的标准接口，BT HAL 的头文件位于如下所示的文件中。

```
hardware/libhardware/include/hardware/bluetooth.h
hardware/libhardware/include/hardware/bt_*.h
```

JNI 中 sBluetoothA2dpInterface 是一个 btav_interface_t 结构体，位于 hardware/libhardware/include/hardware/bt_av.h 中，具体定义代码如下所示。

```
typedef struct {
    size_t          size;
    bt_status_t (*init)( btav_callbacks_t* callbacks );
    bt_status_t (*connect)( bt_bdaddr_t *bd_addr );
    bt_status_t (*disconnect)( bt_bdaddr_t *bd_addr );
    void   (*cleanup)( void );
} btav_interface_t;
```

Android 系统新版本默认蓝牙协议栈 BlueDroid 在如下所示的目录下实现。

```
external/bluetooth/bluedroid
```

上述 stack 实现了通用的 BT HAL 并且也可以通过扩展和改变配置来自定义。例如 A2dp 的连接会调用到 external/bluetooth/bluedroid/btif/src/btif_av.c 的 connect()方法，此方法的具体实现代码如下所示。

```
static bt_status_t connect(bt_bdaddr_t *bd_addr)
{
    BTIF_TRACE_EVENT1("%s", __FUNCTION__);
    CHECK_BTAV_INIT();
    return btif_queue_connect(UUID_SERVCLASS_AUDIO_SOURCE, bd_addr, connect_int);
}
```

第 7 章　NFC 近场通信

NFC 是近场通信（Near Field Communication）的缩写，随着移动智能设备的发展和普及，在很多智能手机中都提供了 NFC 近场通信功能。作为一款著名的操作系统，Android 系统提供了对 NFC 技术的完整支持。本章将详细讲解在 Android 外设项目开发使用近场通信技术的基本知识，为学习本书后面的知识打下基础。

7.1　近场通信技术基础

知识点讲解：光盘:视频\知识点\第 7 章\近场通信技术基础.avi

NFC 近场通信技术是由非接触式射频识别（RFID）及互联互通技术整合演变而来，在单一芯片上结合感应式读卡器、感应式卡片和点对点的功能，能在短距离内与兼容设备进行识别和数据交换。工作频率为 13.56MHz，但是使用这种手机支付方案的用户必须更换特制的手机。目前这项技术在日韩被广泛应用。手机用户凭着配置了支付功能的手机就可以行遍全国：他们的手机可以用作机场登机验证、大厦的门禁钥匙、交通一卡通、信用卡、支付卡等。本节将简要讲解 NFC 技术的基本知识。

7.1.1　NFC 技术的特点

近场通信是基于 RFID 技术发展起来的一种近距离无线通信技术。与 RFID 一样，近场通信信息也是通过频谱中无线频率部分的电磁感应耦合方式传递的，但两者之间还是存在很大的区别。近场通信的传输范围比 RFID 小，RFID 的传输范围可以达到 0～1m，但由于近场通信采取了独特的信号衰减技术，相对于 RFID 来说近场通信具有成本低、带宽高、能耗低等特点。

在现实应用中，近场通信技术主要特征如下。

- 用于近距离（10cm以内）安全通信的无线通信技术。
- 射频频率：13.56MHz。
- 射频兼容：ISO 14443，ISO 15693，Felica标准。
- 数据传输速度：106Kbit/s，212 Kbit/s，424Kbit/s。

7.1.2　NFC 的工作模式

在现实应用中，NFC 技术有如下 3 种工作模式。

- 卡模式（Card emulation）：此模式其实就是相当于一张采用RFID技术的IC卡。可以替代现在大量的IC卡（包括信用卡）场合商场刷卡、公交卡、门禁管制、车票、门票等。此种方式，有一个极大的优点，那就是卡片通过非接触读卡器的 RF 域来供电，即便是寄主设备（如手机）没电也可以工作。

- ❑ 点对点模式（P2P mode）：此模式和红外线差不多，可用于数据交换，只是传输距离较短，传输创建速度较快，传输速度也快，功耗低（蓝牙也类似）。将两个具备NFC功能的设备链接，能实现数据点对点传输，如下载音乐、交换图片或者同步设备地址簿。因此通过NFC，多个设备如数位相机、PDA、计算机和手机之间都可以交换资料或者服务。
- ❑ 读卡器模式（Reader/writer mode）：作为非接触读卡器使用，比如从海报或者展览信息电子标签上读取相关信息。

7.1.3　NFC 和蓝牙的对比

NFC 的最大数据传输量是 424 Kbit/s，这远小于 Bluetooth V2.1（2.1 Mbit/s）。虽然 NFC 在传输速度与距离方面比不上 BlueTooth，但是 NFC 技术不需要电源，对于移动电话或是移动消费性电子产品来说，NFC 的使用比较方便。NFC 的短距离通信特性正是其优点，由于耗电量低、一次只和一台机器连接，拥有较高的保密性与安全性，NFC 有利于信用卡交易时避免被盗用。NFC 的目标并非取代蓝牙等其他无线技术，而是在不同的场合、不同的领域起到相互补充的作用。

NFC 技术和蓝牙技术相比，主要支持功能参数如表 7-1 所示。

表 7-1　NFC 技术和蓝牙技术的参数对比

说　　明	NFC	Bluetooth	Bluetooth Low Energy
RFID 兼容	ISO 18000-3	active	active
标准化机构	ISO/IEC	Bluetooth SIG	Bluetooth SIG
网络标准	ISO 13157 etc.	IEEE 802.15.1	IEEE 802.15.1
网络类型	Point-to-point	WPAN	WPAN
加密	not with RFID	available	available
范围	< 0.2m	~10m (class 2)	~1m (class 3)
频率	13.56 MHz	2.4～2.5 GHz	2.4～2.5 GHz
Bit rate	424 Kbit/s	2.1 Mbit/s	~1.0 Mbit/s
设置程序	< 0.1s	< 6s	< 1s
功耗	< 15mA (read)	varies with class	< 15mA (xmit)

7.2　射频识别技术详解

知识点讲解：光盘:视频\知识点\第 7 章\射频识别技术详解.avi

射频识别即 RFID（Radio Frequency Identification）技术，又称无线射频识别，是 NFC 技术的一个子集。RFID 是一种通信技术，可以通过无线电信号识别特定目标并读写相关数据，而无须识别系统与特定目标之间建立机械或光学接触。在现实中常用的 RFID 技术有低频（125～134.2kHz）、高频（13.56MHz）、超高频，微波等技术。RFID 读写器也分移动式的和固定式的，目前 RFID 技术应用很广，如图书馆、门禁系统和食品安全溯源等。本节将详细讲解射频识别技术 RFID 的基本知识。

7.2.1 RFID 技术简介

从概念上来讲，RFID 类似于条码扫描，对于条码技术而言，它是将已编码的条形码附着于目标物，并使用专用的扫描读写器利用光信号将信息由条形磁传送到扫描读写器；而 RFID 则使用专用的 RFID 读写器及专门的可附着于目标物的 RFID 标签，利用频率信号将信息由 RFID 标签传送至 RFID 读写器。

无线电的信号是通过调成无线电频率的电磁场，把附着在物品上的标签数据传送出去，以自动辨识与追踪该物品。某些标签在识别时从识别器发出的电磁场中就可以得到能量，并不需要电池；也有标签本身拥有电源，并可以主动发出无线电波（调成无线电频率的电磁场）。标签包含了电子存储的信息，数米之内都可以识别。与条形码不同的是，射频标签不需要处在识别器视线之内，也可以嵌入被追踪物体之内。

在现实应用中，有许多行业都运用了射频识别技术。将标签附着在一辆正在生产中的汽车，厂家便可以追踪此车在生产线上的进度。射频标签也可以附于牲畜与宠物上，方便人们对牲畜与宠物的积极识别（积极识别意思是防止数只牲畜使用同一个身份）。射频识别的身份识别卡可以使员工得以进入锁住的建筑部分，汽车上的射频应答器也可以用来征收收费路段与停车场的费用。

某些射频标签附在衣物、个人财物上，甚至植入人体之内。由于这项技术可能会在未经本人许可的情况下读取个人信息，所以这项技术也会有侵犯个人隐私的忧患。

7.2.2 RFID 技术的组成

从结构上讲 RFID 是一种简单的无线系统，只有两个基本器件，该系统用于控制、检测和跟踪物体。系统由一个询问器和很多应答器组成。在最初的技术领域中，应答器是指能够传输信息、回复信息的电子模块。近年来，由于射频技术发展迅猛，应答器有了新的概念和含义，又被叫做智能标签或标签。RFID 电子标签的阅读器通过天线与 RFID 电子标签进行无线通信，可以实现对标签识别码和内存数据的读出或写入操作。RFID 技术可识别高速运动物体并可同时识别多个标签，操作快捷方便。

伴随着 RFID 技术的不断发展，其具体组成如下。

- 应答器：由天线、耦合元件及芯片组成，一般来说都是用标签作为应答器，每个标签具有唯一的电子编码，附着在物体上标识目标对象。
- 阅读器：由天线、耦合元件及芯片组成，读取（有时还可以写入）标签信息的设备，可设计为手持式RFID读写器（如C5000W）或固定式读写器。
- 应用软件系统：是应用层软件，主要是把收集的数据进一步处理，并为人们所使用。

7.2.3 RFID 技术的特点

射频识别系统最重要的优点是非接触识别，它能穿透雪、雾、冰、涂料、尘垢和条形码无法使用的恶劣环境阅读标签，并且阅读速度极快，大多数情况下不到 100ms。有源式射频识别系统的速写能力也是重要的优点，可用于流程跟踪和维修跟踪等交互式业务。

制约射频识别系统发展的主要问题是不兼容的标准。射频识别系统的主要厂商提供的都是专用系统，导致不同的应用和不同的行业采用不同厂商的频率和协议标准，这种混乱和割据的状况已经制约

了整个射频识别行业的增长。许多欧美组织正在着手解决这个问题，并已经取得了一些成绩。标准化必将刺激射频识别技术的大幅度发展和广泛应用。

RFID 技术的主要特点如下。

- 快速扫描：RFID辨识器可以同时辨识读取数个RFID标签。
- 体积小型化、形状多样化：RFID在读取上并不受尺寸大小与形状限制，不需为了读取精确度而配合纸张的固定尺寸和印刷品质。此外，RFID标签更可往小型化与多样形态发展，以应用于不同产品。
- 抗污染能力和耐久性：传统条形码的载体是纸张，因此容易受到污染，但RFID对水、油和化学药品等物质具有很强抵抗性。此外，由于条形码是附于塑料袋或外包装纸箱上，所以特别容易受到折损；RFID卷标是将数据存在芯片中，因此可以免受污损。
- 可重复使用：当今的条形码印刷上去之后就无法更改，RFID标签则可以重复地新增、修改、删除RFID卷标内存储的数据，方便信息的更新。
- 穿透性和无屏障阅读：在被覆盖的情况下，RFID能够穿透纸张、木材和塑料等非金属或非透明的材质，并能够进行穿透性通信。而条形码扫描机必须在近距离而且没有物体阻挡的情况下，才可以辨读条形码。
- 数据的记忆容量大：一维条形码的容量是50B，二维条形码最大的容量可存储2～3000字符，RFID最大的容量则有数兆字节。随着记忆载体的发展，数据容量也有不断扩大的趋势。未来物品所需携带的资料量会越来越大，对卷标所能扩充容量的需求也相应增加。
- 安全性：由于RFID承载的是电子式信息，其数据内容可经由密码保护，使其内容不易被伪造及变造。

RFID 因其所具备的远距离读取、高存储量等特性而备受瞩目。它不仅可以帮助一个企业大幅提高货物、信息管理的效率，还可以让销售企业和制造企业互联，从而更加准确地接收反馈信息，控制需求信息，优化整个供应链。

7.2.4　RFID 技术的工作原理

RFID 技术的基本工作原理是：当标签进入磁场后，接收解读器发出的射频信号，凭借感应电流所获得的能量发送出存储在芯片中的产品信息（Passive Tag，无源标签或被动标签），或者由标签主动发送某一频率的信号（Active Tag，有源标签或主动标签），解读器读取信息并解码后，送至中央信息系统进行有关数据处理。

一套完整的 RFID 系统，是由阅读器（Reader）与电子标签（TAG）也就是所谓的应答器（Transponder）及应用软件系统 3 个部分所组成，其工作原理是 Reader 发射一特定频率的无线电波能量给 Transponder，用以驱动 Transponder 电路将内部的数据送出，此时 Reader 便依序接收解读数据，送给应用程序做相应的处理。

以 RFID 卡片阅读器及电子标签之间的通信及能量感应方式来看，可以大致将 RFID 分成感应耦合（Inductive Coupling）及后向散射耦合（Backscatter Coupling）两种。通常低频的 RFID 大都采用第一种方式，而较高频大多采用第二种方式。

阅读器根据使用的结构和技术不同可以是读或读/写装置，是 RFID 系统信息控制和处理中心。阅读器通常由耦合模块、收发模块、控制模块和接口单元组成。阅读器和应答器之间一般采用半双工通

信方式进行信息交换，同时阅读器通过耦合给无源应答器提供能量和时序。在实际应用中，可进一步通过 Ethernet 或 WLAN 等实现对物体识别信息的采集、处理及远程传送等管理功能。应答器是 RFID 系统的信息载体，应答器大多是由耦合原件（线圈、微带天线等）和微芯片组成无源单元。

7.3 Android 系统中的 NFC

知识点讲解：光盘:视频\知识点\第 7 章\Android 系统中的 NFC.avi

NFC 通信总是由一个发起者（Initiator）和一个接受者（Target）组成，通常 Initiator 主动发送电磁场（RF）可以为被动式接受者（Passive Target）提供电源，其工作的基本原理和收音机类似。正是由于被动式接受者可以通过发起者提供电源，因此 Target 可以有非常简单的形式，比如标签、卡和 Sticker 的形式。另外，NFC 也支持点到点的通信（Peer to Peer），此时参与通信的双方都有电源支持。在 Android 系统的 NFC 模块应用中，Android 手机通常是作为通信中的发起者，也就是作为 NFC 的读写器。Android 手机也可以模拟作为 NFC 通信的接受者，并且从 Android 2.3.3 起也支持 P2P 通信。Android 系统支持如下的 NFC 标准。

- NfcANFC-A (ISO 14443-3A)
- NfcBNFC-B (ISO 14443-3B)
- NfcFNFC-F (JIS 6319-4)
- NfcVNFC-V (ISO 15693)
- IsoDepISO-DEP (ISO 14443-4)
- MifareClassic
- MifareUltralight

在 Android 系统中，NFC 模块从上到下的结构如图 7-1 所示。

```
-------------------- /system/framework/framework.jar---------------------

android.nfc                标准接口 （NFCAdapter/NfcManager）
android.nfc.tech           标签技术

---------------------------- /system/Nfc.apk-----------------------------

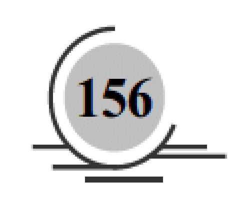
com.android.nfc             NFC 服务相关
          .DeviceHost               底层设备接口原型
          .NfcService               Nfc 服务 实现 DeviceHostListener 接口
com.android.nfc.dhimpl      NFC 功能底层实现-com.android.nfc.DeviceHost (NXP)
          .NativeNfcManager         implements DeviceHost
          JNI-> com_android_nfc_NativeNfcManager.cpp (libnfc_jni.so)
          .NativeNfcSecureElement
          JNI-> com_android_nfc_NativeNfcSecureElement.cpp (libnfc_jni.so)

----------------------- /system/lib/libnfc___.so-------------------------

libnfc-nxp => libnfc.so, libnfc_ndef.so
libnfc-nci => libnfc-nci.so
```

图 7-1 NFC 模块从上到下的结构

在本节的内容中，将详细讲解 Android 系统中 NFC 模块源码的基本知识。



7.3.1　分析 Java 层

在 Android 系统中，NFC 模块的 Java 层代码位于如下所示的目录中。

```
\frameworks\base\core\java\android\nfc\
```

在上述目录中，包含了用来与本地 NFC 适配器进行交互的顶层类，这些类可以表示被检测到的 tags 和用 NDEF 数据格式。在 NFC 的 Java 层中，常用的顶层类如下。

（1）NfcManager 类

类 NfcManager 在文件\frameworks\base\core\java\android\nfc\NfcManager.java 中定义，这是一个 NFC Adapter 的管理器，可以列出所有此 Android 设备支持的 NFC Adapter，只不过大部分 Android 设备只有一个 NFC Adapter，所以在大部分情况下可以直接用静态方法 getDefaultAdapter(context)来取适配器。文件\frameworks\base\core\java\android\nfc\NfcManager.java 的具体实现代码如下所示。

```
public final class NfcManager {
    private final NfcAdapter mAdapter;

    /**
     * @hide
     */
    public NfcManager(Context context) {
        NfcAdapter adapter;
        context = context.getApplicationContext();
        if (context == null) {
            throw new IllegalArgumentException(
                    "context not associated with any application (using a mock context?)");
        }
        try {
            adapter = NfcAdapter.getNfcAdapter(context);
        } catch (UnsupportedOperationException e) {
            adapter = null;
        }
        mAdapter = adapter;
    }

    /**
     * Get the default NFC Adapter for this device.
     *
     * @return the default NFC Adapter
     */
    public NfcAdapter getDefaultAdapter() {
        return mAdapter;
    }
}
```

（2）NfcAdapter 类

类 NfcAdapter 在文件\frameworks\base\core\java\android\nfc\NfcAdapter.java 中定义，此类表示本设备的 NFC Adapter，可以定义 Intent 来请求将系统检测到 tags 的提醒发送到我们的 Activity，并提供方法去注册前台 tag 提醒发布和前台 NDEF 推送。 前台 NDEF 推送是当前 Android 版本唯一支持的 p2p

NFC 通信方式。文件\frameworks\base\core\java\android\nfc\NfcAdapter.java 的具体实现代码如下所示。

```
public final class NfcAdapter {
    static final String TAG = "NFC";
    public static final String ACTION_NDEF_DISCOVERED = "android.nfc.action.NDEF_DISCOVERED";
    @SdkConstant(SdkConstantType.ACTIVITY_INTENT_ACTION)
    public static final String ACTION_TECH_DISCOVERED = "android.nfc.action.TECH_DISCOVERED";
    @SdkConstant(SdkConstantType.ACTIVITY_INTENT_ACTION)
    public static final String ACTION_TAG_DISCOVERED = "android.nfc.action.TAG_DISCOVERED";
    public static final String ACTION_TAG_LEFT_FIELD = "android.nfc.action.TAG_LOST";
    public static final String EXTRA_TAG = "android.nfc.extra.TAG";
    public static final String EXTRA_NDEF_MESSAGES = "android.nfc.extra.NDEF_MESSAGES";
    public static final String EXTRA_ID = "android.nfc.extra.ID";
    @SdkConstant(SdkConstantType.BROADCAST_INTENT_ACTION)
    public static final String ACTION_ADAPTER_STATE_CHANGED =
            "android.nfc.action.ADAPTER_STATE_CHANGED";
    public static final String EXTRA_ADAPTER_STATE = "android.nfc.extra.ADAPTER_STATE";

    public static final int STATE_OFF = 1;
    public static final int STATE_TURNING_ON = 2;
    public static final int STATE_ON = 3;
    public static final int STATE_TURNING_OFF = 4;

    /** @hide */
    public static final int FLAG_NDEF_PUSH_NO_CONFIRM = 0x1;

    /** @hide */
    public static final String ACTION_HANDOVER_TRANSFER_STARTED =
            "android.nfc.action.HANDOVER_TRANSFER_STARTED";

    /** @hide */
    public static final String ACTION_HANDOVER_TRANSFER_DONE =
            "android.nfc.action.HANDOVER_TRANSFER_DONE";

    /** @hide */
    public static final String EXTRA_HANDOVER_TRANSFER_STATUS =
            "android.nfc.extra.HANDOVER_TRANSFER_STATUS";

    /** @hide */
    public static final int HANDOVER_TRANSFER_STATUS_SUCCESS = 0;
    /** @hide */
    public static final int HANDOVER_TRANSFER_STATUS_FAILURE = 1;

    /** @hide */
    public static final String EXTRA_HANDOVER_TRANSFER_URI =
            "android.nfc.extra.HANDOVER_TRANSFER_URI";

    // Guarded by NfcAdapter.class
    static boolean sIsInitialized = false;

    // Final after first constructor, except for
```

```
// attemptDeadServiceRecovery() when NFC crashes - we accept a best effort
// recovery
static INfcAdapter sService;
static INfcTag sTagService;
static HashMap<Context, NfcAdapter> sNfcAdapters = new HashMap(); //guard by NfcAdapter.class
static NfcAdapter sNullContextNfcAdapter;   // protected by NfcAdapter.class

final NfcActivityManager mNfcActivityManager;
final Context mContext;
public interface OnNdefPushCompleteCallback {
    public void onNdefPushComplete(NfcEvent event);
}
public interface CreateNdefMessageCallback {
    public NdefMessage createNdefMessage(NfcEvent event);
}
// TODO javadoc
public interface CreateBeamUrisCallback {
    public Uri[ ] createBeamUris(NfcEvent event);
}
private static boolean hasNfcFeature() {
    IPackageManager pm = ActivityThread.getPackageManager();
    if (pm == null) {
        Log.e(TAG, "Cannot get package manager, assuming no NFC feature");
        return false;
    }
    try {
        return pm.hasSystemFeature(PackageManager.FEATURE_NFC);
    } catch (RemoteException e) {
        Log.e(TAG, "Package manager query failed, assuming no NFC feature", e);
        return false;
    }
}
public static synchronized NfcAdapter getNfcAdapter(Context context) {
    if (!sIsInitialized) {
        /* is this device meant to have NFC */
        if (!hasNfcFeature()) {
            Log.v(TAG, "this device does not have NFC support");
            throw new UnsupportedOperationException();
        }

        sService = getServiceInterface();
        if (sService == null) {
            Log.e(TAG, "could not retrieve NFC service");
            throw new UnsupportedOperationException();
        }
        try {
            sTagService = sService.getNfcTagInterface();
        } catch (RemoteException e) {
            Log.e(TAG, "could not retrieve NFC Tag service");
            throw new UnsupportedOperationException();
        }
```

```
            sIsInitialized = true;
        }
        if (context == null) {
            if (sNullContextNfcAdapter == null) {
                sNullContextNfcAdapter = new NfcAdapter(null);
            }
            return sNullContextNfcAdapter;
        }
        NfcAdapter adapter = sNfcAdapters.get(context);
        if (adapter == null) {
            adapter = new NfcAdapter(context);
            sNfcAdapters.put(context, adapter);
        }
        return adapter;
    }

    /** get handle to NFC service interface */
    private static INfcAdapter getServiceInterface() {
        /* get a handle to NFC service */
        IBinder b = ServiceManager.getService("nfc");
        if (b == null) {
            return null;
        }
        return INfcAdapter.Stub.asInterface(b);
    }
    public static NfcAdapter getDefaultAdapter(Context context) {
        if (context == null) {
            throw new IllegalArgumentException("context cannot be null");
        }
        context = context.getApplicationContext();
        if (context == null) {
            throw new IllegalArgumentException(
                    "context not associated with any application (using a mock context?)");
        }
        /* use getSystemService() for consistency */
        NfcManager manager = (NfcManager) context.getSystemService(Context.NFC_SERVICE);
        if (manager == null) {
            // NFC not available
            return null;
        }
        return manager.getDefaultAdapter();
    }
    @Deprecated
    public static NfcAdapter getDefaultAdapter() {
        // introduced in API version 9 (GB 2.3)
        // deprecated in API version 10 (GB 2.3.3)
        // removed from public API in version 16 (ICS MR2)
        // should maintain as a hidden API for binary compatibility for a little longer
        Log.w(TAG, "WARNING: NfcAdapter.getDefaultAdapter() is deprecated, use " +
                "NfcAdapter.getDefaultAdapter(Context) instead", new Exception());
```

```
        return NfcAdapter.getNfcAdapter(null);
    }
    public boolean isEnabled() {
        try {
            return sService.getState() == STATE_ON;
        } catch (RemoteException e) {
            attemptDeadServiceRecovery(e);
            return false;
        }
    }
    public int getAdapterState() {
        try {
            return sService.getState();
        } catch (RemoteException e) {
            attemptDeadServiceRecovery(e);
            return NfcAdapter.STATE_OFF;
        }
    }
    public boolean enable() {
        try {
            return sService.enable();
        } catch (RemoteException e) {
            attemptDeadServiceRecovery(e);
            return false;
        }
    }
    public boolean disable() {
        try {
            return sService.disable(true);
        } catch (RemoteException e) {
            attemptDeadServiceRecovery(e);
            return false;
        }
    }

    public void setBeamPushUris(Uri[ ] uris, Activity activity) {
        if (activity == null) {
            throw new NullPointerException("activity cannot be null");
        }
        if (uris != null) {
            for (Uri uri : uris) {
                if (uri == null) throw new NullPointerException("Uri not " +
                        "allowed to be null");
                String scheme = uri.getScheme();
                if (scheme == null || (!scheme.equalsIgnoreCase("file") &&
                        !scheme.equalsIgnoreCase("content"))) {
                    throw new IllegalArgumentException("URI needs to have " +
                            "either scheme file or scheme content");
                }
            }
```

```
        }
        mNfcActivityManager.setNdefPushContentUri(activity, uris);
    }
    public void setBeamPushUrisCallback(CreateBeamUrisCallback callback, Activity activity) {
        if (activity == null) {
            throw new NullPointerException("activity cannot be null");
        }
        mNfcActivityManager.setNdefPushContentUriCallback(activity, callback);
    }
    public void setNdefPushMessage(NdefMessage message, Activity activity,
            Activity ... activities) {
        int targetSdkVersion = getSdkVersion();
        try {
            if (activity == null) {
                throw new NullPointerException("activity cannot be null");
            }
            mNfcActivityManager.setNdefPushMessage(activity, message, 0);
            for (Activity a : activities) {
                if (a == null) {
                    throw new NullPointerException("activities cannot contain null");
                }
                mNfcActivityManager.setNdefPushMessage(a, message, 0);
            }
        } catch (IllegalStateException e) {
            if (targetSdkVersion < android.os.Build.VERSION_CODES.JELLY_BEAN) {
                // Less strict on old applications - just log the error
                Log.e(TAG, "Cannot call API with Activity that has already " +
                        "been destroyed", e);
            } else {
                // Prevent new applications from making this mistake, re-throw
                throw(e);
            }
        }
    }

    /**
     * @hide
     */
    public void setNdefPushMessage(NdefMessage message, Activity activity, int flags) {
        if (activity == null) {
            throw new NullPointerException("activity cannot be null");
        }
        mNfcActivityManager.setNdefPushMessage(activity, message, flags);
    }
    public void setNdefPushMessageCallback(CreateNdefMessageCallback callback, Activity activity,
            Activity ... activities) {
        int targetSdkVersion = getSdkVersion();
        try {
            if (activity == null) {
                throw new NullPointerException("activity cannot be null");
            }
```

```
            mNfcActivityManager.setNdefPushMessageCallback(activity, callback, 0);
            for (Activity a : activities) {
                if (a == null) {
                    throw new NullPointerException("activities cannot contain null");
                }
                mNfcActivityManager.setNdefPushMessageCallback(a, callback, 0);
            }
        } catch (IllegalStateException e) {
            if (targetSdkVersion < android.os.Build.VERSION_CODES.JELLY_BEAN) {
                // Less strict on old applications - just log the error
                Log.e(TAG, "Cannot call API with Activity that has already " +
                        "been destroyed", e);
            } else {
                // Prevent new applications from making this mistake, re-throw
                throw(e);
            }
        }
    }
    public void setNdefPushMessageCallback(CreateNdefMessageCallback callback, Activity activity,
            int flags) {
        if (activity == null) {
            throw new NullPointerException("activity cannot be null");
        }
        mNfcActivityManager.setNdefPushMessageCallback(activity, callback, flags);
    }
    public void setOnNdefPushCompleteCallback(OnNdefPushCompleteCallback callback,
            Activity activity, Activity ... activities) {
        int targetSdkVersion = getSdkVersion();
        try {
            if (activity == null) {
                throw new NullPointerException("activity cannot be null");
            }
            mNfcActivityManager.setOnNdefPushCompleteCallback(activity, callback);
            for (Activity a : activities) {
                if (a == null) {
                    throw new NullPointerException("activities cannot contain null");
                }
                mNfcActivityManager.setOnNdefPushCompleteCallback(a, callback);
            }
        } catch (IllegalStateException e) {
            if (targetSdkVersion < android.os.Build.VERSION_CODES.JELLY_BEAN) {
                // Less strict on old applications - just log the error
                Log.e(TAG, "Cannot call API with Activity that has already " +
                        "been destroyed", e);
            } else {
                // Prevent new applications from making this mistake, re-throw
                throw(e);
            }
        }
    }
    public void enableForegroundDispatch(Activity activity, PendingIntent intent,
```

```
                IntentFilter[ ] filters, String[ ][ ] techLists) {
        if (activity == null || intent == null) {
            throw new NullPointerException();
        }
        if (!activity.isResumed()) {
            throw new IllegalStateException("Foreground dispatch can only be enabled " +
                    "when your activity is resumed");
        }
        try {
            TechListParcel parcel = null;
            if (techLists != null && techLists.length > 0) {
                parcel = new TechListParcel(techLists);
            }
            ActivityThread.currentActivityThread().registerOnActivityPausedListener(activity,
                    mForegroundDispatchListener);
            sService.setForegroundDispatch(intent, filters, parcel);
        } catch (RemoteException e) {
            attemptDeadServiceRecovery(e);
        }
    }
    public void disableForegroundDispatch(Activity activity) {
        ActivityThread.currentActivityThread().unregisterOnActivityPausedListener(activity,
                mForegroundDispatchListener);
        disableForegroundDispatchInternal(activity, false);
    }
```

（3）NdefMessage 和 NdefRecord 类

NDEF 是 NFC 论坛定义的数据结构，用来将有效的存数据存储到 NFC tags 中，比如文本、URL 和其他 MIME 类型。一个 NdefMessage 扮演一个容器，这个容器存储那些发送和读到的数据。一个 NdefMessage 对象包含 0 或多个 NdefRecord，每个 NDEF record 有一个类型，比如文本、URL、智慧型海报/广告，或其他 MIME 数据。在 NdefMessage 中的第一个 NfcRecord 的类型，功能是发送 tag 到一个 Android 设备上的 Activity。其中类 NdefMessage 在文件\frameworks\base\core\java\android\nfc\NdefMessage.java 中定义，具体实现代码如下所示。

```
public final class NdefMessage implements Parcelable {
    private final NdefRecord[ ] mRecords;
    public NdefMessage(byte[ ] data) throws FormatException {
        if (data == null) throw new NullPointerException("data is null");
        ByteBuffer buffer = ByteBuffer.wrap(data);

        mRecords = NdefRecord.parse(buffer, false);

        if (buffer.remaining() > 0) {
            throw new FormatException("trailing data");
        }
    }
    public NdefMessage(NdefRecord record, NdefRecord ... records) {
        // validate
        if (record == null) throw new NullPointerException("record cannot be null");

        for (NdefRecord r : records) {
```

```
            if (r == null) {
                throw new NullPointerException("record cannot be null");
            }
        }

        mRecords = new NdefRecord[1 + records.length];
        mRecords[0] = record;
        System.arraycopy(records, 0, mRecords, 1, records.length);
    }
    public NdefMessage(NdefRecord[ ] records) {
        // validate
        if (records.length < 1) {
            throw new IllegalArgumentException("must have at least one record");
        }
        for (NdefRecord r : records) {
            if (r == null) {
                throw new NullPointerException("records cannot contain null");
            }
        }

        mRecords = records;
    }
    public NdefRecord[ ] getRecords() {
        return mRecords;
    }
    public int getByteArrayLength() {
        int length = 0;
        for (NdefRecord r : mRecords) {
            length += r.getByteLength();
        }
        return length;
    }
    public byte[ ] toByteArray() {
        int length = getByteArrayLength();
        ByteBuffer buffer = ByteBuffer.allocate(length);

        for (int i=0; i<mRecords.length; i++) {
            boolean mb = (i == 0);    // first record
            boolean me = (i == mRecords.length - 1);    // last record
            mRecords[i].writeToByteBuffer(buffer, mb, me);
        }

        return buffer.array();
    }

    @Override
    public int describeContents() {
        return 0;
    }
```

```
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeInt(mRecords.length);
        dest.writeTypedArray(mRecords, flags);
    }

    public static final Parcelable.Creator<NdefMessage> CREATOR =
            new Parcelable.Creator<NdefMessage>() {
        @Override
        public NdefMessage createFromParcel(Parcel in) {
            int recordsLength = in.readInt();
            NdefRecord[ ] records = new NdefRecord[recordsLength];
            in.readTypedArray(records, NdefRecord.CREATOR);
            return new NdefMessage(records);
        }
        @Override
        public NdefMessage[ ] newArray(int size) {
            return new NdefMessage[size];
        }
    };

    @Override
    public int hashCode() {
        return Arrays.hashCode(mRecords);
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null) return false;
        if (getClass() != obj.getClass()) return false;
        NdefMessage other = (NdefMessage) obj;
        return Arrays.equals(mRecords, other.mRecords);
    }

    @Override
    public String toString() {
        return "NdefMessage " + Arrays.toString(mRecords);
    }
}
```

（4）Tag 类

类 Tag 在文件\frameworks\base\core\java\android\nfc\Tag.java 中定义，此类用于标示一个被动的 NFC 目标，比如 tag、card 和钥匙挂扣，甚至是一个电话模拟的 NFC 卡。当一个 tag 被检测到时，一个 tag 对象将被创建并且封装到一个 Intent 里，然后 NFC 发布系统将这个 Intent 用 startActivity 发送到注册了接受这种 Intent 的 Activity 里。我们可以用方法 getTechList()来得到这个 tag 支持的技术细节，并创建一个 android.nfc.tech 提供的相应的 TagTechnology 对象。文件 Tag.java 的具体实现代码如下所示。

```
public final class Tag implements Parcelable {
    final byte[ ] mId;
    final int[ ] mTechList;
```

```
final String[ ] mTechStringList;
final Bundle[ ] mTechExtras;
final int mServiceHandle;    // for use by NFC service, 0 indicates a mock
final INfcTag mTagService; // interface to NFC service, will be null if mock tag

int mConnectedTechnology;
public Tag(byte[ ] id, int[ ] techList, Bundle[ ] techListExtras, int serviceHandle,
        INfcTag tagService) {
    if (techList == null) {
        throw new IllegalArgumentException("rawTargets cannot be null");
    }
    mId = id;
    mTechList = Arrays.copyOf(techList, techList.length);
    mTechStringList = generateTechStringList(techList);
    // Ensure mTechExtras is as long as mTechList
    mTechExtras = Arrays.copyOf(techListExtras, techList.length);
    mServiceHandle = serviceHandle;
    mTagService = tagService;

    mConnectedTechnology = -1;
}
public static Tag createMockTag(byte[ ] id, int[ ] techList, Bundle[ ] techListExtras) {
    // set serviceHandle to 0 and tagService to null to indicate mock tag
    return new Tag(id, techList, techListExtras, 0, null);
}
```

（5）“tech\” 子目录

除此之外，在“frameworks\base\core\java\android\nfc\tech\”目录中包含了查询 tag 属性和进行 I/O 操作的类。这些类分别表示一个 tag 支持的不同的 NFC 技术标准，如图 7-2 所示。

frameworks ▸ base ▸ core ▸ java ▸ android ▸ nfc ▸ tech　　搜索 tech

享 ▾　刻录　新建文件夹

名称	修改日期	类型	大小
BasicTagTechnology.java	2013/8/14 9:20	JAVA 文件	5 KB
IsoDep.java	2013/8/14 9:20	JAVA 文件	8 KB
MifareClassic.java	2013/8/14 9:20	JAVA 文件	24 KB
MifareUltralight.java	2013/8/14 9:20	JAVA 文件	11 KB
Ndef.java	2013/8/14 9:20	JAVA 文件	15 KB
NdefFormatable.java	2013/8/14 9:20	JAVA 文件	7 KB
NfcA.java	2013/8/14 9:20	JAVA 文件	6 KB
NfcB.java	2013/8/14 9:20	JAVA 文件	5 KB
NfcBarcode.java	2013/8/14 9:20	JAVA 文件	5 KB
NfcF.java	2013/8/14 9:20	JAVA 文件	6 KB
NfcV.java	2013/8/14 9:20	JAVA 文件	5 KB
package.html	2013/8/14 9:20	HTML 文档	1 KB
TagTechnology.java	2013/8/14 9:20	JAVA 文件	9 KB

图 7-2　“frameworks\base\core\java\android\nfc\tech\”目录

在图 7-2 所示的目录中，类 TagTechnology 是如表 7-2 中所有 TagTechnology 类必须实现的。

表 7-2 必须实现 TagTechnology 的类

类	说　明
NfcA	支持 ISO 14443-3A 标准的操作。Provides access to NFC-A (ISO 14443-3A) properties and I/O operations
NfcB	Provides access to NFC-B (ISO 14443-3B) properties and I/O operations
NfcF	Provides access to NFC-F (JIS 6319-4) properties and I/O operations
NfcV	Provides access to NFC-V (ISO 15693) properties and I/O operations
IsoDep	Provides access to ISO-DEP (ISO 14443-4) properties and I/O operations
Ndef	提供对那些被格式化为 NDEF 的 tag 的数据的访问和其他操作

而类 NdefFormatable 对那些可以被格式化成 NDEF 格式的 tag 提供一个格式化的操作，文件 frameworks\base\core\java\android\nfc\tech\NdefFormatable.java 的具体实现代码如下所示。

```
public final class NdefFormatable extends BasicTagTechnology {
    private static final String TAG = "NFC";
    public static NdefFormatable get(Tag tag) {
        if (!tag.hasTech(TagTechnology.NDEF_FORMATABLE)) return null;
        try {
            return new NdefFormatable(tag);
        } catch (RemoteException e) {
            return null;
        }
    }
    /*package*/ void format(NdefMessage firstMessage, boolean makeReadOnly) throws IOException,
            FormatException {
        checkConnected();

        try {
            int serviceHandle = mTag.getServiceHandle();
            INfcTag tagService = mTag.getTagService();
            int errorCode = tagService.formatNdef(serviceHandle, MifareClassic.KEY_DEFAULT);
            switch (errorCode) {
                case ErrorCodes.SUCCESS:
                    break;
                case ErrorCodes.ERROR_IO:
                    throw new IOException();
                case ErrorCodes.ERROR_INVALID_PARAM:
                    throw new FormatException();
                default:
                    // Should not happen
                    throw new IOException();
            }
            // Now check and see if the format worked
            if (!tagService.isNdef(serviceHandle)) {
                throw new IOException();
            }

            // Write a message, if one was provided
            if (firstMessage != null) {
                errorCode = tagService.ndefWrite(serviceHandle, firstMessage);
                switch (errorCode) {
```

```
                    case ErrorCodes.SUCCESS:
                        break;
                    case ErrorCodes.ERROR_IO:
                        throw new IOException();
                    case ErrorCodes.ERROR_INVALID_PARAM:
                        throw new FormatException();
                    default:
                        // Should not happen
                        throw new IOException();
                }
            }

            // optionally make read-only
            if (makeReadOnly) {
                errorCode = tagService.ndefMakeReadOnly(serviceHandle);
                switch (errorCode) {
                    case ErrorCodes.SUCCESS:
                        break;
                    case ErrorCodes.ERROR_IO:
                        throw new IOException();
                    case ErrorCodes.ERROR_INVALID_PARAM:
                        throw new IOException();
                    default:
                        // Should not happen
                        throw new IOException();
                }
            }
        } catch (RemoteException e) {
            Log.e(TAG, "NFC service dead", e);
        }
    }
}
```

类 MifareClassic 在文件 frameworks\base\core\java\android\nfc\tech\MifareClassic.java 中定义，如果 Android 设备支持 MIFARE，则提供对 MIFARE Classic 目标的属性和 I/O 操作。文件 MifareClassic.java 的具体实现代码如下所示。

```
public final class MifareClassic extends BasicTagTechnology {
    private static final String TAG = "NFC";
    public static final byte[ ] KEY_DEFAULT =
            {(byte)0xFF,(byte)0xFF,(byte)0xFF,(byte)0xFF,(byte)0xFF,(byte)0xFF};
    public static final byte[ ] KEY_MIFARE_APPLICATION_DIRECTORY =
            {(byte)0xA0,(byte)0xA1,(byte)0xA2,(byte)0xA3,(byte)0xA4,(byte)0xA5};
    public static final byte[ ] KEY_NFC_FORUM =
            {(byte)0xD3,(byte)0xF7,(byte)0xD3,(byte)0xF7,(byte)0xD3,(byte)0xF7};

    /** A MIFARE Classic compatible card of unknown type */
    public static final int TYPE_UNKNOWN = -1;
    /** A MIFARE Classic tag */
    public static final int TYPE_CLASSIC = 0;
    /** A MIFARE Plus tag */
    public static final int TYPE_PLUS = 1;
```

```
/** A MIFARE Pro tag */
public static final int TYPE_PRO = 2;

/** Tag contains 16 sectors, each with 4 blocks. */
public static final int SIZE_1K = 1024;
/** Tag contains 32 sectors, each with 4 blocks. */
public static final int SIZE_2K = 2048;
/**
 * Tag contains 40 sectors. The first 32 sectors contain 4 blocks and the last 8 sectors
 * contain 16 blocks.
 */
public static final int SIZE_4K = 4096;
/** Tag contains 5 sectors, each with 4 blocks. */
public static final int SIZE_MINI = 320;

/** Size of a MIFARE Classic block (in bytes) */
public static final int BLOCK_SIZE = 16;

private static final int MAX_BLOCK_COUNT = 256;
private static final int MAX_SECTOR_COUNT = 40;

private boolean mIsEmulated;
private int mType;
private int mSize;
public static MifareClassic get(Tag tag) {
    if (!tag.hasTech(TagTechnology.MIFARE_CLASSIC)) return null;
    try {
        return new MifareClassic(tag);
    } catch (RemoteException e) {
        return null;
    }
}
/** @hide */
public MifareClassic(Tag tag) throws RemoteException {
    super(tag, TagTechnology.MIFARE_CLASSIC);
    NfcA a = NfcA.get(tag);  // MIFARE Classic is always based on NFC a
    mIsEmulated = false;
    switch (a.getSak()) {
    case 0x01:
    case 0x08:
        mType = TYPE_CLASSIC;
        mSize = SIZE_1K;
        break;
    case 0x09:
        mType = TYPE_CLASSIC;
        mSize = SIZE_MINI;
        break;
    case 0x10:
        mType = TYPE_PLUS;
        mSize = SIZE_2K;
        // SecLevel = SL2
```

```
            break;
        case 0x11:
            mType = TYPE_PLUS;
            mSize = SIZE_4K;
            // Seclevel = SL2
            break;
        case 0x18:
            mType = TYPE_CLASSIC;
            mSize = SIZE_4K;
            break;
        case 0x28:
            mType = TYPE_CLASSIC;
            mSize = SIZE_1K;
            mIsEmulated = true;
            break;
        case 0x38:
            mType = TYPE_CLASSIC;
            mSize = SIZE_4K;
            mIsEmulated = true;
            break;
        case 0x88:
            mType = TYPE_CLASSIC;
            mSize = SIZE_1K;
            // NXP-tag: false
            break;
        case 0x98:
        case 0xB8:
            mType = TYPE_PRO;
            mSize = SIZE_4K;
            break;
        default:
            // Stack incorrectly reported a MifareClassic. We cannot handle this
            // gracefully - we have no idea of the memory layout. Bail.
            throw new RuntimeException(
                    "Tag incorrectly enumerated as MIFARE Classic, SAK = " + a.getSak());
        }
    }
```

类 MifareUltralight 在文件 frameworks\base\core\java\android\nfc\tech\MifareUltralight.java 中定义，如果 Android 设备支持 MIFARE，则提供对 MIFARE Ultralight 目标的属性和 I/O 操作。

```
public final class MifareUltralight extends BasicTagTechnology {
    private static final String TAG = "NFC";
    /** A MIFARE Ultralight compatible tag of unknown type */
    public static final int TYPE_UNKNOWN = -1;
    /** A MIFARE Ultralight tag */
    public static final int TYPE_ULTRALIGHT = 1;
    /** A MIFARE Ultralight C tag */
    public static final int TYPE_ULTRALIGHT_C = 2;
    /** Size of a MIFARE Ultralight page in bytes */
    public static final int PAGE_SIZE = 4;
    private static final int NXP_MANUFACTURER_ID = 0x04;
```

```
private static final int MAX_PAGE_COUNT = 256;
/** @hide */
public static final String EXTRA_IS_UL_C = "isulc";
private int mType;
public static MifareUltralight get(Tag tag) {
    if (!tag.hasTech(TagTechnology.MIFARE_ULTRALIGHT)) return null;
    try {
        return new MifareUltralight(tag);
    } catch (RemoteException e) {
        return null;
    }
}
/** @hide */
public MifareUltralight(Tag tag) throws RemoteException {
    super(tag, TagTechnology.MIFARE_ULTRALIGHT);
    // Check if this could actually be a MIFARE
    NfcA a = NfcA.get(tag);
    mType = TYPE_UNKNOWN;
    if (a.getSak() == 0x00 && tag.getId()[0] == NXP_MANUFACTURER_ID) {
        Bundle extras = tag.getTechExtras(TagTechnology.MIFARE_ULTRALIGHT);
        if (extras.getBoolean(EXTRA_IS_UL_C)) {
            mType = TYPE_ULTRALIGHT_C;
        } else {
            mType = TYPE_ULTRALIGHT;
        }
    }
}
public byte[ ] readPages(int pageOffset) throws IOException {
    validatePageIndex(pageOffset);
    checkConnected();

    byte[ ] cmd = { 0x30, (byte) pageOffset};
    return transceive(cmd, false);
}
public void writePage(int pageOffset, byte[ ] data) throws IOException {
    validatePageIndex(pageOffset);
    checkConnected();
    byte[ ] cmd = new byte[data.length + 2];
    cmd[0] = (byte) 0xA2;
    cmd[1] = (byte) pageOffset;
    System.arraycopy(data, 0, cmd, 2, data.length);
    transceive(cmd, false);
}
public void setTimeout(int timeout) {
    try {
        int err = mTag.getTagService().setTimeout(
                TagTechnology.MIFARE_ULTRALIGHT, timeout);
        if (err != ErrorCodes.SUCCESS) {
            throw new IllegalArgumentException("The supplied timeout is not valid");
        }
    } catch (RemoteException e) {
```

```
                Log.e(TAG, "NFC service dead", e);
            }
        }
        public int getTimeout() {
            try {
                return mTag.getTagService().getTimeout(TagTechnology.MIFARE_ULTRALIGHT);
            } catch (RemoteException e) {
                Log.e(TAG, "NFC service dead", e);
                return 0;
            }
        }

        private static void validatePageIndex(int pageIndex) {
            // Do not be too strict on upper bounds checking, since some cards
            // may have more addressable memory than they report.
            // Note that issuing a command to an out-of-bounds block is safe - the
            // tag will wrap the read to an addressable area. This validation is a
            // helper to guard against obvious programming mistakes.
            if (pageIndex < 0 || pageIndex >= MAX_PAGE_COUNT) {
                throw new IndexOutOfBoundsException("page out of bounds: " + pageIndex);
            }
        }
}
```

7.3.2　分析 JNI 部分

在 Android 系统中，NFC 模块的 JNI 部分代码通过如下所示的目录中实现。

```
\packages\apps\Nfc\nxp\jni
```

JNI 部分代码向上会跟 Framework 层的 Java 代码进行交互，向下会跟 libnfc 层进行交互。JNI 层的核心文件是 com_android_nfc_NativeNfcManager.cpp，功能是初始化并启动 NFC 服务，并扫描和读取 tag。文件 com_android_nfc_NativeNfcManager.cpp 的主要实现代码如下所示。

```
static void client_kill_deferred_call(void* arg)
{
   struct nfc_jni_native_data *nat = (struct nfc_jni_native_data *)arg;

   nat->running = FALSE;
}

static void kill_client(nfc_jni_native_data *nat)
{
   phDal4Nfc_Message_Wrapper_t   wrapper;
   phLibNfc_DeferredCall_t *pMsg;

   usleep(50000);

   ALOGD("Terminating client thread...");

   pMsg = (phLibNfc_DeferredCall_t*)malloc(sizeof(phLibNfc_DeferredCall_t));
   pMsg->pCallback = client_kill_deferred_call;
```

```
    pMsg->pParameter = (void*)nat;

    wrapper.msg.eMsgType        = PH_LIBNFC_DEFERREDCALL_MSG;
    wrapper.msg.pMsgData        = pMsg;
    wrapper.msg.Size            = sizeof(phLibNfc_DeferredCall_t);

    phDal4Nfc_msgsnd(gDrvCfg.nClientId, (struct msgbuf *)&wrapper, sizeof(phLibNfc_Message_t), 0);
}

static void nfc_jni_ioctl_callback(void *pContext, phNfc_sData_t *pOutput, NFCSTATUS status) {
    struct nfc_jni_callback_data * pCallbackData = (struct nfc_jni_callback_data *) pContext;
    LOG_CALLBACK("nfc_jni_ioctl_callback", status);

    /* Report the callback status and wake up the caller */
    pCallbackData->status = status;
    sem_post(&pCallbackData->sem);
}

static void nfc_jni_deinit_download_callback(void *pContext, NFCSTATUS status)
{
    struct nfc_jni_callback_data * pCallbackData = (struct nfc_jni_callback_data *) pContext;
    LOG_CALLBACK("nfc_jni_deinit_download_callback", status);

    /* Report the callback status and wake up the caller */
    pCallbackData->status = status;
    sem_post(&pCallbackData->sem);
}

static int nfc_jni_download_locked(struct nfc_jni_native_data *nat, uint8_t update)
{
     uint8_t OutputBuffer[1];
     uint8_t InputBuffer[1];
     struct timespec ts;
     NFCSTATUS status = NFCSTATUS_FAILED;
     phLibNfc_StackCapabilities_t caps;
     struct nfc_jni_callback_data cb_data;
     bool result;

     /* Create the local semaphore */
     if (!nfc_cb_data_init(&cb_data, NULL))
     {
         goto clean_and_return;
     }

     if(update)
     {
          //deinit
          TRACE("phLibNfc_Mgt_DeInitialize() (download)");
          REENTRANCE_LOCK();
          status = phLibNfc_Mgt_DeInitialize(gHWRef, nfc_jni_deinit_download_callback, (void *)&cb_data);
```

```
        REENTRANCE_UNLOCK();
        if (status != NFCSTATUS_PENDING)
        {
            ALOGE("phLibNfc_Mgt_DeInitialize() (download) returned 0x%04x[%s]", status, nfc_jni_get_
status_name(status));
        }

        clock_gettime(CLOCK_REALTIME, &ts);
        ts.tv_sec += 5;

        /* Wait for callback response */
        if(sem_timedwait(&cb_data.sem, &ts))
        {
            ALOGW("Deinitialization timed out (download)");
        }

        if(cb_data.status != NFCSTATUS_SUCCESS)
        {
            ALOGW("Deinitialization FAILED (download)");
        }
        TRACE("Deinitialization SUCCESS (download)");
    }

    result = performDownload(nat, false);

    if (!result) {
        status = NFCSTATUS_FAILED;
        goto clean_and_return;
    }

    TRACE("phLibNfc_Mgt_Initialize()");
    REENTRANCE_LOCK();
    status = phLibNfc_Mgt_Initialize(gHWRef, nfc_jni_init_callback, (void *)&cb_data);
    REENTRANCE_UNLOCK();
    if(status != NFCSTATUS_PENDING)
    {
        ALOGE("phLibNfc_Mgt_Initialize() (download) returned 0x%04x[%s]", status, nfc_jni_get_status_name(status));
        goto clean_and_return;
    }
    TRACE("phLibNfc_Mgt_Initialize() returned 0x%04x[%s]", status, nfc_jni_get_status_name(status));

    if(sem_wait(&cb_data.sem))
    {
        ALOGE("Failed to wait for semaphore (errno=0x%08x)", errno);
        status = NFCSTATUS_FAILED;
        goto clean_and_return;
    }

    /* Initialization Status */
    if(cb_data.status != NFCSTATUS_SUCCESS)
```

```
    {
        status = cb_data.status;
        goto clean_and_return;
    }

    /* ====== CAPABILITIES ======= */
    REENTRANCE_LOCK();
    status = phLibNfc_Mgt_GetstackCapabilities(&caps, (void*)nat);
    REENTRANCE_UNLOCK();
    if (status != NFCSTATUS_SUCCESS)
    {
      ALOGW("phLibNfc_Mgt_GetstackCapabilities returned 0x%04x[%s]", status, nfc_jni_get_status_name(status));
    }
    else
    {
        ALOGD("NFC capabilities: HAL = %x, FW = %x, HW = %x, Model = %x, HCI = %x, Full_FW = %d, Rev
= %d, FW Update Info = %d",
              caps.psDevCapabilities.hal_version,
              caps.psDevCapabilities.fw_version,
              caps.psDevCapabilities.hw_version,
              caps.psDevCapabilities.model_id,
              caps.psDevCapabilities.hci_version,
              caps.psDevCapabilities.full_version[NXP_FULL_VERSION_LEN-1],
              caps.psDevCapabilities.full_version[NXP_FULL_VERSION_LEN-2],
              caps.psDevCapabilities.firmware_update_info);
    }

    /*Download is successful*/
    status = NFCSTATUS_SUCCESS;

clean_and_return:
   nfc_cb_data_deinit(&cb_data);
   return status;
}

static int nfc_jni_configure_driver(struct nfc_jni_native_data *nat)
{
    char value[PROPERTY_VALUE_MAX];
    int result = FALSE;
    NFCSTATUS status;

    /* ====== CONFIGURE DRIVER ======= */
    /* Configure hardware link */
    gDrvCfg.nClientId = phDal4Nfc_msgget(0, 0600);

    TRACE("phLibNfc_Mgt_ConfigureDriver(0x%08x)", gDrvCfg.nClientId);
    REENTRANCE_LOCK();
    status = phLibNfc_Mgt_ConfigureDriver(&gDrvCfg, &gHWRef);
    REENTRANCE_UNLOCK();
    if(status == NFCSTATUS_ALREADY_INITIALISED) {
           ALOGW("phLibNfc_Mgt_ConfigureDriver() returned 0x%04x[%s]", status, nfc_jni_get_status_name(status));
```

```
    }
    else if(status != NFCSTATUS_SUCCESS)
    {
        ALOGE("phLibNfc_Mgt_ConfigureDriver() returned 0x%04x[%s]", status, nfc_jni_get_status_name(status));
        goto clean_and_return;
    }
    TRACE("phLibNfc_Mgt_ConfigureDriver() returned 0x%04x[%s]", status, nfc_jni_get_status_name(status));

    if(pthread_create(&(nat->thread), NULL, nfc_jni_client_thread, nat) != 0)
    {
        ALOGE("pthread_create failed");
        goto clean_and_return;
    }

    driverConfigured = TRUE;

clean_and_return:
    return result;
}

static int nfc_jni_unconfigure_driver(struct nfc_jni_native_data *nat)
{
    int result = FALSE;
    NFCSTATUS status;

    /* Unconfigure driver */
    TRACE("phLibNfc_Mgt_UnConfigureDriver()");
    REENTRANCE_LOCK();
    status = phLibNfc_Mgt_UnConfigureDriver(gHWRef);
    REENTRANCE_UNLOCK();
    if(status != NFCSTATUS_SUCCESS)
    {
        ALOGE("phLibNfc_Mgt_UnConfigureDriver() returned error 0x%04x[%s] -- this should never happen",
status, nfc_jni_get_status_name( status));
    }
    else
    {
        ALOGD("phLibNfc_Mgt_UnConfigureDriver() returned 0x%04x[%s]", status, nfc_jni_get_status_name
(status));
        result = TRUE;
    }

    driverConfigured = FALSE;

    return result;
}
```

7.3.3　分析底层

在 Android 系统中，NFC 模块的底层部分有驱动部分和 libnfc 库部分两大类。驱动部分在 device

目录中实现，例如 device\samsung\tuna\nfc 目录中保存了设备厂家三星电子提供的 hardware lib。而 libnfc-nci 和 libnfc-nxp 目录中的底层文件则负责 NFC 数据的读取和解析工作。

7.4 在 Android 系统编写 NFC APP 的方法

知识点讲解：光盘:视频\知识点\第 7 章\在 Android 系统编写 NFC APP 的方法.avi

当 Android 手机开启了 NFC 程序，并且检测到一个 TAG 后，TAG 分发系统会自动创建一个封装了 NFC TAG 信息的 Intent。如果多于一个应用程序能够处理这个 Intent，那么手机就会弹出一个对话框，让用户选择处理该 TAG 的 Activity。在 TAG 分发系统中定义了 3 种 Intent，按优先级从高到低排列顺序依次是：

- NDEF_DISCOVERED
- TECH_DISCOVERED
- TAG_DISCOVERED

当 Android 设备检测到有 NFC Tag 靠近时，会根据 Action 申明的顺序给对应的 Activity 发送含 NFC 消息的 Intent。此处我们使用的 intent-filter 的 Action 类型为 TECH_DISCOVERED，从而可以处理所有类型为 ACTION_TECH_DISCOVERED，并且使用的技术为 nfc_tech_filter.xml 文件中定义的类型的 TAG。

当 Android 手机检测到一个 TAG 时，启用 Activity 的匹配过程如图 7-3 所示。

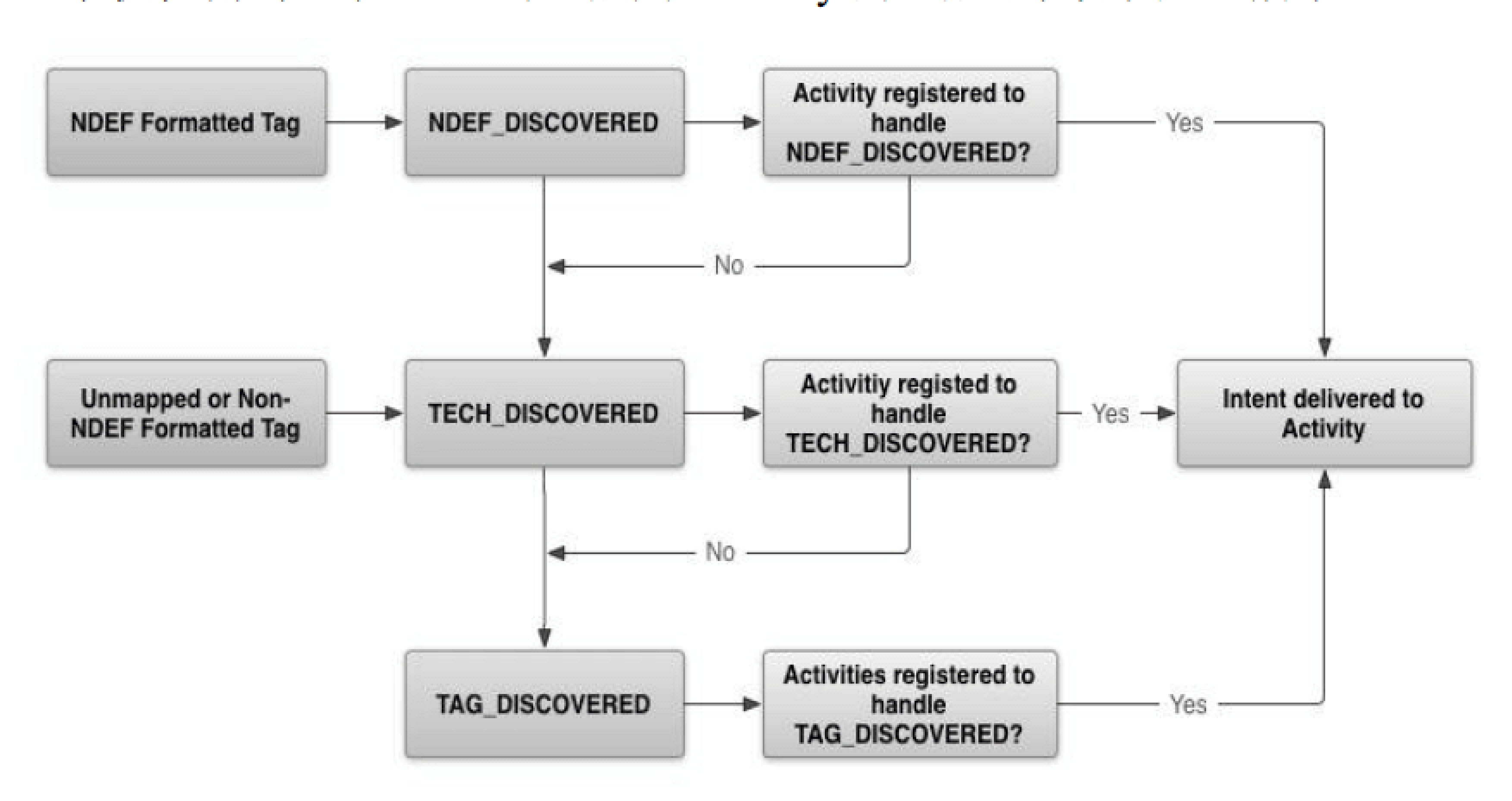


图 7-3　启用 Activity 的匹配过程

在 Android 系统中，编写 NFC APP 的基本流程如下。

（1）在相关的 androidManifest 文件中设置 NFC 权限，具体代码如下所示。

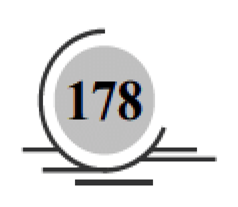
```
<uses-permission android:name="android.permission.NFC" />
```

（2）设置 SDK 的级别限制，例如设置为 API 10。

```
<uses-sdk android:minSdkVersion="10"/>
```

（3）声明特殊功能的限制权限，通过如下声明可以让应用程序在 Google Play 上声明使用者必须拥有 NFC 功能。

```
<uses-feature android:name="android.hardware.nfc" android:required="true" />
```



（4）实现 NFC 标签过滤。

在 Activity 的 Intent 过滤 XML 声明中，可以同时声明过滤如下 3 种 action（动作），但是需要提前知道系统在发送 Intent 时拥有的优先级。

- ❑ 动作1：过滤ACTION_TAG_DISCOVERED。

```
<intent-filter>
    <action android:name="android.nfc.action.TAG_DISCOVERED"/>
    <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
```

这个最简单，也是最后一个被尝试接受 Intent 的选项。

- ❑ 动作2：过滤ACTION_NDEF_DISCOVERED。

```
<intent-filter>
<action android:name="android.nfc.action.NDEF_DISCOVERED"/>
<category android:name="android.intent.category.DEFAULT"/>
<data android:mimeType="text/plain" />
 </intent-filter>
```

其中最重要的是 data 的 mimeType 类型，此定义越准确，Intent 指向这个 Activity 的成功率就越高，否则系统可能不会发出你想要的 NDEF intent 了。

- ❑ 动作3：过滤ACTION_TECH_DISCOVERED。

首先需要在<project-path>/res/xml 下面创建一个过滤规则文件，可以任意命名，例如可以叫做 nfc_tech_filter.xml。这个里面定义的是 NFC 实现的各种标准，每一个 NFC 卡都会符合多个不同的标准。可以在检测到 NFC 标签后，使用 getTechList()方法来查看所检测的 tag 到底支持哪些 NFC 标准。在一个 nfc_tech_filter.xml 文件中可以定义多个<tech-list>结构组，每一组代表声明只接受同时满足这些标准的 NFC 标签。比如 A 组表示，只有同时满足 IsoDep、NfcA、NfcB、NfcF 这 4 个标准的 NFC 标签的 Intent 才能进入。A 与 B 组之间的关系就是只要满足其中一个就可以了。换句话说，我们的 NFC 标签技术满足 A 的声明也可以，满足 B 的声明也可以。

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
<tech-list> -----------------------------A 组
<tech>android.nfc.tech.IsoDep</tech> <tech>android.nfc.tech.NfcA</tech>   <tech>android.nfc.tech.NfcB</tech>
<tech>android.nfc.tech.NfcF</tech>
</tech-list>
<tech-list>------------------------------------B 组
<tech>android.nfc.tech.NfcV</tech> <tech>android.nfc.tech.Ndef</tech> <tech>android.nfc.tech.NdefFormatable</tech>
 <tech>android.nfc.tech.MifareClassic</tech> <tech>android.nfc.tech.MifareUltralight</tech>
 </tech-list>
</resources>
```

在 androidManifest 文件中，声明 xml 过滤的举例代码如下所示。

```
<activity>
 <intent-filter>
<action android:name="android.nfc.action.TECH_DISCOVERED"/>
</intent-filter>
<meta-data android:name="android.nfc.action.TECH_DISCOVERED"
      android:resource="@xml/nfc_tech_filter" />------------这个就是你的资源文件名
 </activity>
```

（5）创建 NFC 标签的前台分发系统。

什么是 NFC 的前台发布系统？就是说当已经打开我们的应用的时候，那么通过这个前台发布系统

的设置，可以让已经启动的 Activity 拥有更高的优先级来依据在代码中定义的标准过滤和处理 Intent，而不是让其他声明了 Intent Filter 的 Activity 来干扰，甚至连自己声明在文件 androidManifest 中的 Intent Filter 都不会来干扰。也就是说，foreground Dispatch 的优先级大于 intent filter。此时有如下两种情况。

- 第一种情况：当Activity没有启动的时候去扫描tag，那么系统中所有的Intent Filter都将一起参与过滤。
- 第二种情况：当Activity启动去扫描tag时，那么将直接使用在foreground dispatch中代码写入的过滤标准。如果这个标准没有命中任何Intent，那么系统将使用所有Activity声明的Intent Filter xml来过滤。

例如，在 OnCreate 中可以添加如下所示的代码。

```
// Create a generic PendingIntent that will be deliver to this activity. The NFC stack will fill in the intent with the details of the discovered tag before delivering to this activity.
   mPendingIntent = PendingIntent.getActivity(this, 0,
              new Intent(this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
        //做一个 IntentFilter 过滤你想要的 action 这里过滤的是 ndef
        IntentFilter ndef = new IntentFilter(NfcAdapter.ACTION_NDEF_DISCOVERED);
//如果对 action 的定义有更高的要求，比如 data 的要求，可以使用如下的代码来定义 intentFilter
//        try {
//              ndef.addDataType("*/*");
//        } catch (MalformedMimeTypeException e) {

//              // TODO Auto-generated catch block

//              e.printStackTrace();

//        }
        //生成 intentFilter
        mFilters = new IntentFilter[ ] {
                ndef,
        };
        //做一个 tech-list。可以看到是二维数据，每一个一维数组之间的关系是或，但是一个一维数组之内的各个项就是与的关系了
        mTechLists = new String[ ][ ] {
                new String[ ] { NfcF.class.getName()},
                new String[ ]{NfcA.class.getName()},
                new String[ ]{NfcB.class.getName()},
                new String[ ]{NfcV.class.getName()}
                };
```

在 onPause 和 onResume 中需要加入如下相应的代码。

```
public void onPause() {
 super.onPause();
//反注册 mAdapter.disableForegroundDispatch(this);
}
public void onResume() {
 super.onResume();
//设定 intentfilter 和 tech-list
//如果两个都为 null 就代表优先接收任何形式的 TAG action
//也就是说系统会主动发 TAG intent
mAdapter.enableForegroundDispatch(this, mPendingIntent, mFilters, mTechLists);
}
```

第 8 章　Google Now 和 Android Wear 详解

Google Now 是谷歌在 I/O 开发者大会上随安卓 4.1 系统同时推出的一款应用，它会全面了解你的各种习惯和正在进行的动作，并利用它所了解的来为你提供相关信息。本章将详细讲解在 Android 设备中使用 Google Now 技术的基本知识，为学习本书后面的知识打下基础。

8.1　Google Now 介绍

知识点讲解：光盘:视频\知识点\第 8 章\Google Now 介绍.avi

Google Now 是 Google 在移动市场最重要的创新之一。通过对用户数据的挖掘，Google Now 在适当的时刻提供适当的信息，而它的卡片式推送也代表了 Google 展现信息的新方向。正如 GigaOm 的作者在某次旅行中体会到的，Google Now 成了一个有力的帮手。虽然它仍有些让人不安，但 Google Now 利大于弊。

8.1.1　搜索引擎的升级——Google Now

Google Now 功能是 I/O 大会上的一个亮点，它可以根据不同使用习惯来帮用户进行多项信息的预测，虽然人机交互方面与 iOS 上的 Siri 还有很大差距，但其预测比起 Siri 更加实用。国外媒体都给了 Google Now 功能很高的评价，不过这个功能在中国受到很大的限制。

下面是 Reddit 网站网友对 Google Now 的精彩评论：这是笔者与 Google Now 共处的第一天，当早上醒来时会惊奇地发现 Google Now 居然直接告诉了笔者去兼职工作的路上所要花费的时间。更有趣的是，笔者在手机中保存的“工作地点”其实是笔者的学校，而不是笔者真正工作的地方，笔者只能认为是 Google Now 发现了笔者每个周日都会去那个地方上班吧。

在过去的 10 年中，搜索引擎的核心是获取足够多的海量信息，搜索技术的发展过程是追赶如何更好地获取信息的过程，核心是个性化和实时信息。但是随着时代的进步和发展，现在搜索结果正在变得越来越个性化。不同的人都会看到他更感兴趣的搜索结果，提高了搜索的效率，甚至由于搜索变得过于个性化，人们获得的信息都是自己想看到的，从而让原本能够扩大人们视野的搜索变成了把人们限制在自我的世界工具。这还引发了关于搜索过分个性化可能引发的弊端的讨论。

搜索在个性化方面的努力最重要的是将搜索和社交网络结合，这样搜索引擎就能获得用户的更多信息，从而更好地帮用户作出判断。在个性化搜索方面谷歌遇到了来自 Facebook 的挑战，拥有最多用户信息的网站是 Facebook，但它却并不向谷歌开放。从某种程度上说，谷歌推出自己的社交网络 Google+ 的核心也是希望获得更多的用户信息。

实时搜索更多是搜索在技术实现上的改进，当然，大部分实时信息都存在于 Twitter 和雅虎，这对谷歌也是不小的挑战。随着移动互联网的发展，位置也成为了搜索引擎提供结果的重要依据，这也是

个性化的一部分。而随着位置信息的加入，围绕这一点可以打造一个生活服务的平台。

综上所述，本地搜索将是一个巨大的市场，这时搜索提供的已经不仅仅是信息，更应该是一种服务。正因如此，Google Now 登上了历史舞台，Google Now 还有什么功能？

- ❑ 新的应用会更加方便用户收取电子邮件，当接收到新邮件时，它就会自动弹出以便查看。
- ❑ 实现了办理登记手续的QR CODE终端的更新，但是这一功能目前仅限于美国联合航空公司使用。
- ❑ 具有新的镜头搜索功能，令搜索和查找更加方便准确。
- ❑ 具有步行和行车里程记录功能，这个计步器功能可通过 Android 设备的传感器来统计用户每月行驶的里程，包括步行和骑自行车的路程。
- ❑ 拥有并强化了对博物馆、电影院、餐厅等搜索帮助。
- ❑ 旅游和娱乐特色功能：包括汽车租赁、演唱会门票和通勤共享方面的卡片。公共交通和电视节目的卡片进行了改善，这些卡片现在可以听音识别音乐和节目信息。用户可以为新媒体节目的开播设定搜索提醒，同时还可以接收实时NCAA橄榄球比分。

8.1.2 Google Now 的用法

其实 Google Now 并不是如同 Google Mail、Google Talk 那样的独立 App，Google Now 被 Google 集成到了 Google 搜索中。在正常情况下，开启 Google 搜索即可使用 Google Now。但是因为 Google 搜索业务已经退出中国大陆，所以 Google 也没打算让 Google Now 覆盖中国大陆用户。即使顺利安装了 Google 搜索，依然找不到 Google Now 功能，如图 8-1 所示。

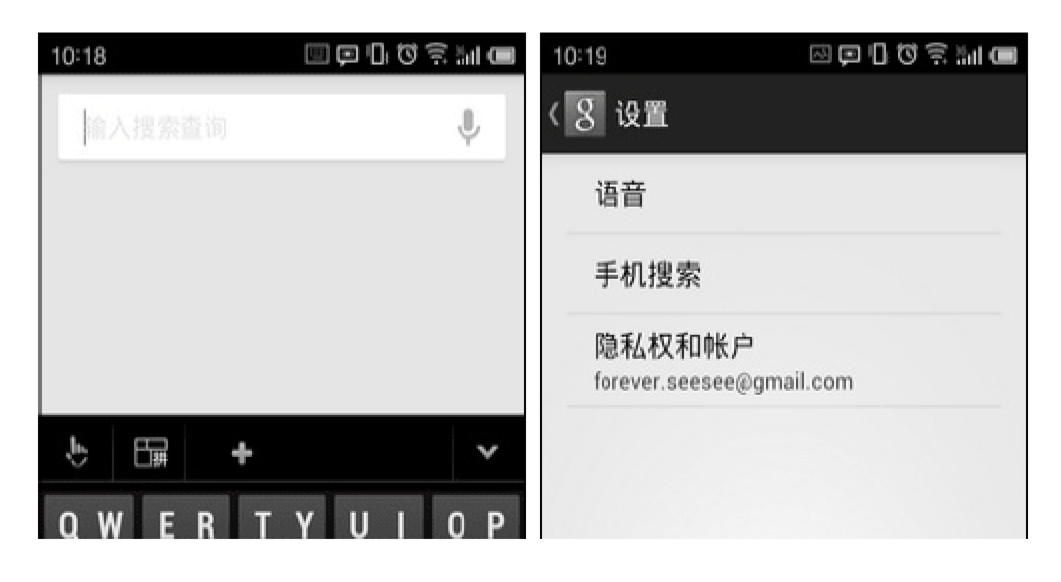


图 8-1　默认没有 Google Now 功能的 Google 搜索

此时需要经过如下所示的步骤进行设置。

（1）登录手机设备的 Google 账户。

（2）在“设置”选项中将系统语言改为英文，如图 8-2 所示。

（3）再次开启 Google Search 后会发现出现 Google Now 了，如图 8-3 所示。



图 8-2　设置设备语言为英文

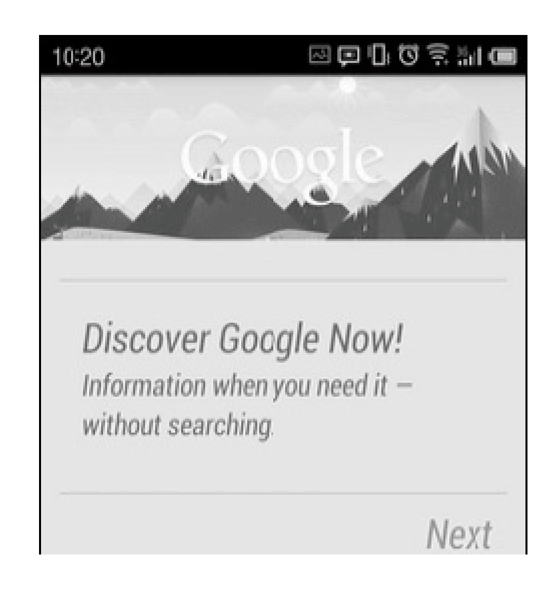
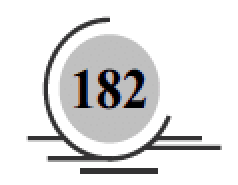


图 8-3　Google Search 中出现 Google Now



（4）按照提示，点击 Next 按钮即可完成 Google Now 的初始化，这时候我们就可以使用 Google Now 了，如图 8-4 所示。

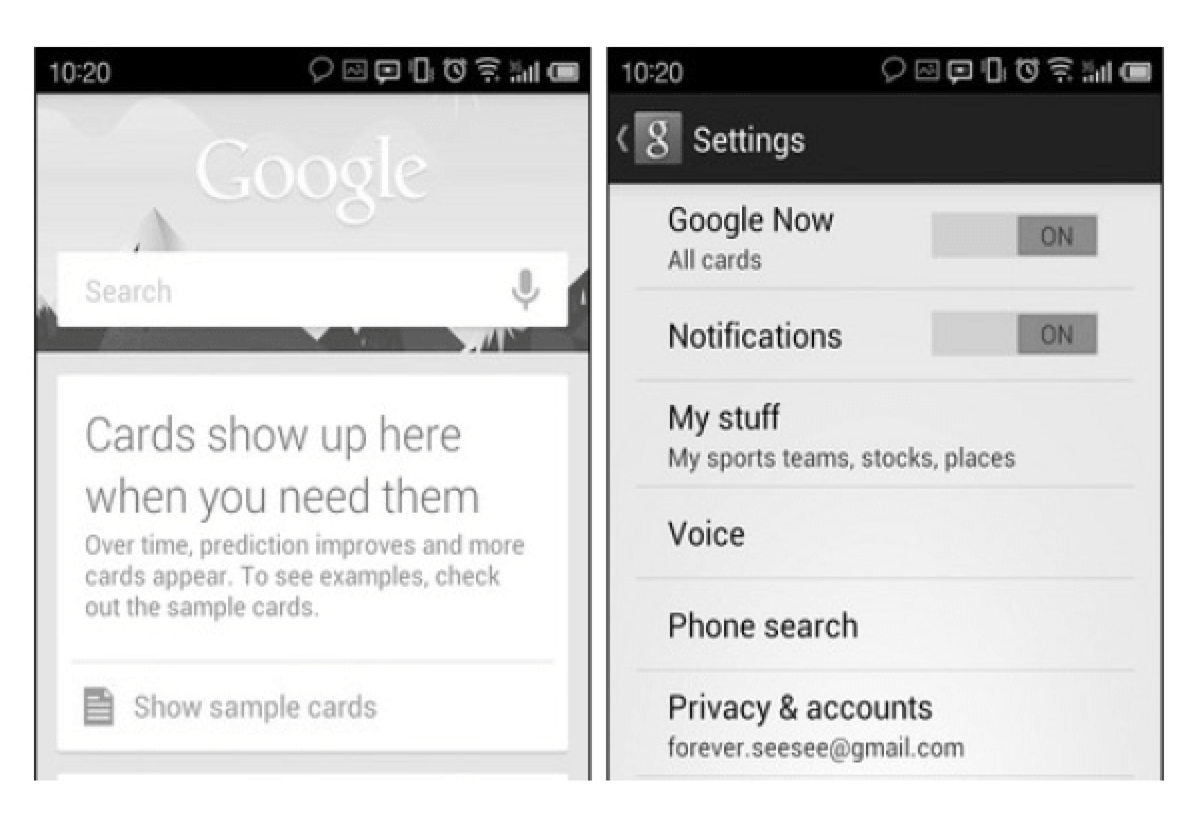


图 8-4　此时可以使用 Google Now

（5）当设置完 Google Now 后回到设置菜单，将系统语言重新设置为简体中文。设置完毕后，Google Now 非但不会被关闭，语言也变成了简体中文。这意味着 Google 本来就做好了 Google Now 的简体中文语言支持，只是没对简体中文用户开放而已，如图 8-5 所示。



图 8-5　中文 Google Now

（6）经过测试后会发现，虽然 Google Now 没有针对国内用户开放，但是数据依然涵盖了国内。在使用期间，公交班次、天气等信息都准确无误，连接也没遇到什么阻碍，如图 8-6 所示。

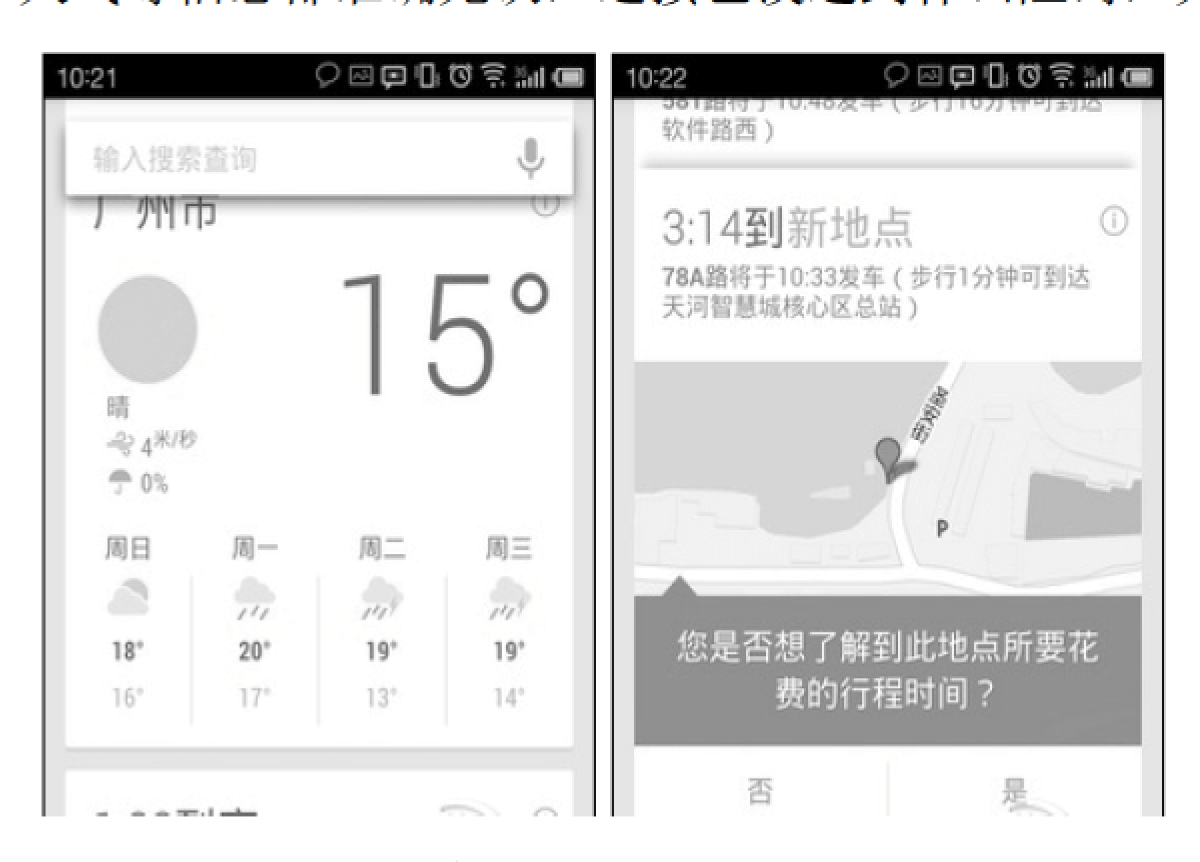


图 8-6　使用 Google Now 的界面

注意：只有在设备中登录并绑定Google账号后才能使用Google Now功能，国产行货手机没有内置添加Google账号功能，读者需要在获取Root权限后进行添加设置。

8.2　Android Wear 详解

知识点讲解：光盘:视频\知识点\第 8 章\Android Wear 详解.avi

2014 年 3 月，继谷歌眼镜之后，谷歌推出了 Android Wear 可穿戴平台，正式进军智能手表领域。与之前传闻不同的是，谷歌并未推出硬件，这意味着什么？显然，作为一个平台服务商，谷歌的目标不仅仅是一款卖得好的智能手表，而是一统整个穿戴式计算机行业。对于用户而言，Android Wear 将改变目前智能手表领域缺乏标准、各自为营的混乱状况，同时也能够与自己的 Android 手机获得更无缝化的数据共享。在本节的内容中，将简单看一看 Android Wear 平台给我们带来了怎样的前景和未来。

8.2.1　什么是 Android Wear

可以将 Android Wear 看作是一个针对智能手表等可穿戴设备优化的 Android 版本，Android Wear 界面更适合小屏幕，主要功能是面向手机与手表互联带来的新型移动体验。举例子来说，平常乘坐公交车时难免会遇到坐过站的情况，只要你在 Android Wear 手表中设定好目的地，GPS 便会开始定位，及时提醒我们拟到达的车站，这样就能够避免发生坐过站的情况。

从本章前面讲解的内容可知，Google Now 应用一直致力于通过上下文联想技术提供全面、智能的搜索体验，现在 Google Now 被集成到 Android Wear 中了，不需要任何按键，只需说 OK，Google 以及你想知道的内容或是进行的操作即可。

谷歌在视频中演示了相当丰富的使用场景，比如你要去海滩冲浪，Android Wear 手表会自动弹出“海里有海蜇”的警告；在收到短信场景时，可以直接语音回复即可；在登机场景中，直接出示手表中的机票二维码就可以完成登机工作。

另外，健身应用也是 Android Wear 必备的一个功能。Android Wear 能够实时监测我们的活动状态，记录步数及热量消耗。当然，健身功能实际上还有很大的发展空间，相信谷歌和手表制造商会在日后为用户提供更多样化的健康监测形式，如手表背面内置传感器监测用户体温和心率等。

由此可见，Android Wear 是将 Android 延伸到可穿戴设备的项目。这个项目首先从智能手表开始。通过一系列的新设备和应用，Android Wear 将能够做到：

- 在你最需要的时候给出有用的信息：从你最喜欢的社交应用中获取更新，使用通信应用交流，从购物应用、新闻应用那里获取通知等。
- 直接回答你的问题：说一声OK Google来提出问题，比如鳄梨里有多少卡路里，航班离开的时间，游戏的分数，或者完成某件事情，比如呼叫出租车、发短信、预订餐厅或者设置闹钟。
- 更好地监控你的健康：通过 Android Wear 上的提醒和健康信息，达到自己健身的目标。你最爱的健身应用能够提供实时的速度、距离和时间信息。
- 通向多屏世界的钥匙：Android Wear 能让你控制其他设备。用OK Google打开手机上的音乐列表，或者将最喜欢的电影投射到电视上面。在开发者的参与下，还会有更多的可能性。

目前，摩托罗拉和 LG 已经展示了概念的 Android Wear 手表，预计三星、HTC、华硕等厂商都会

后续跟进。首先来看看摩托罗拉的 Moto 360 手表，它拥有一个接近传统手表的圆形金属表盘，适合在所有场合佩戴。摩托罗拉公司也承诺将使用精良的材质，保持佩戴的舒适性，如图 8-7 所示。



图 8-7　Android Wear 手表

8.2.2　搭建 Android Wear 开发环境

现在谷歌已经公开了 Android Wear 的预览版，只面向谷歌账号开发者用户公开。具体信息请登录 http://developer.android.com/wear/index.html，如图 8-8 所示。

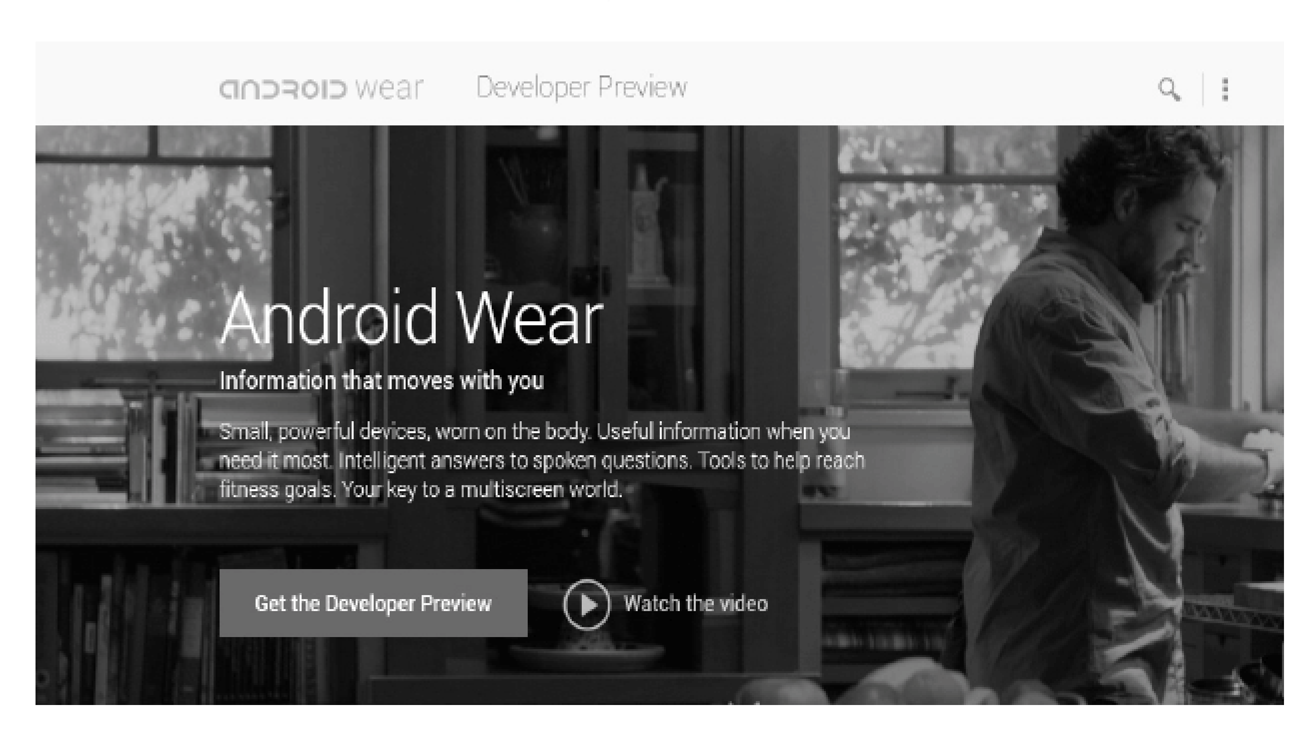


图 8-8　Android Wear 官方站点

单击图 8-8 的 Get Developer Preview 按钮后来到 Android Wear 开发者预览界面，在此列出了搭建开发环境的方法和开发资料，如图 8-9 所示。

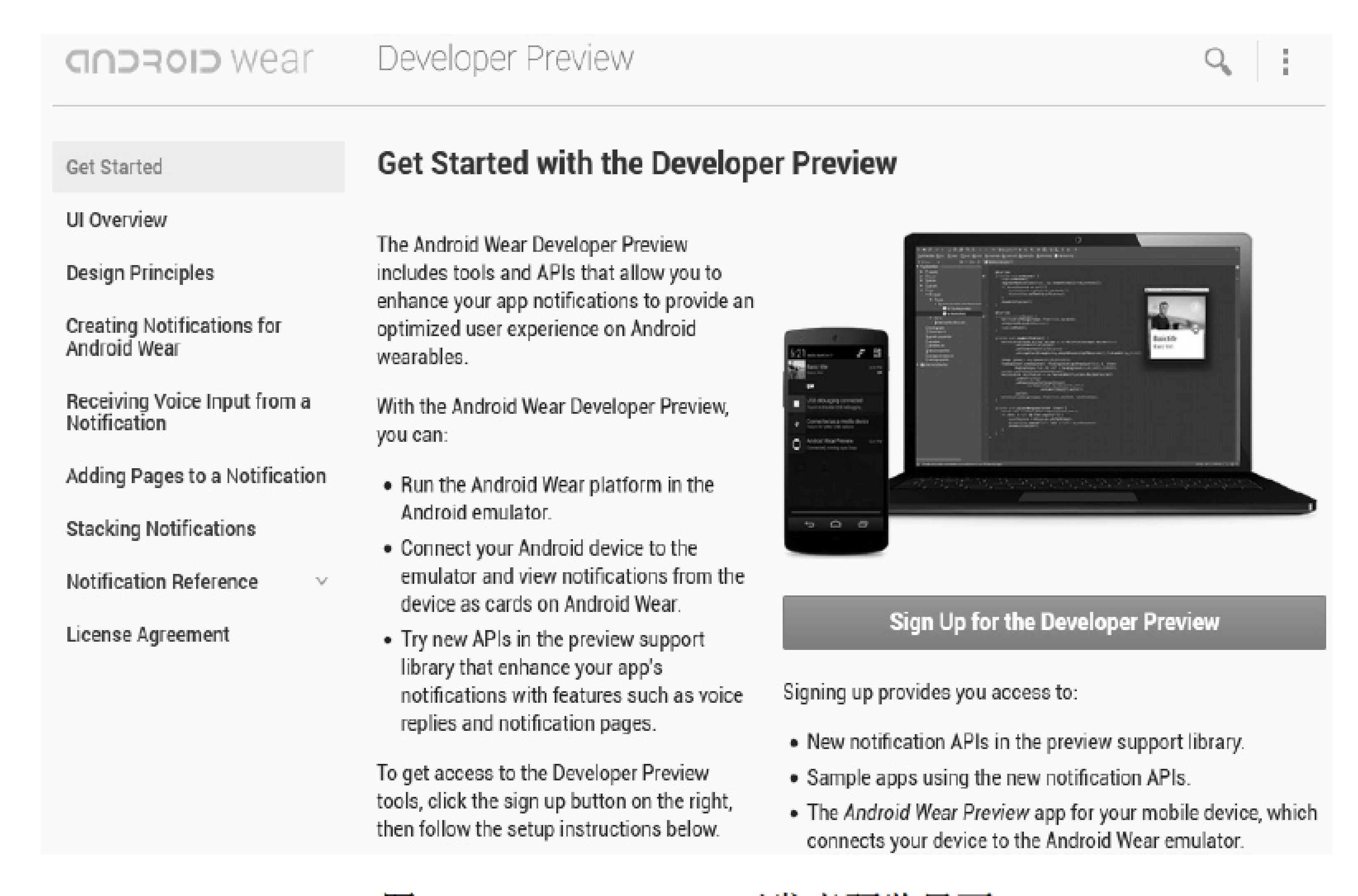


图 8-9　Android Wear 开发者预览界面

Android Wear 开发环境和 Android 应用开发环境类似，具体过程如下所示。

（1）根据本书第 2 章的内容安装 Android SDK，在 Android SDK 中包括了 Android Wear 的所有开发工具。

（2）单击图 8-9 中的按钮来到注册界面，在此界面注册为 Android Wear 预览开发者，如图 8-10 所示。

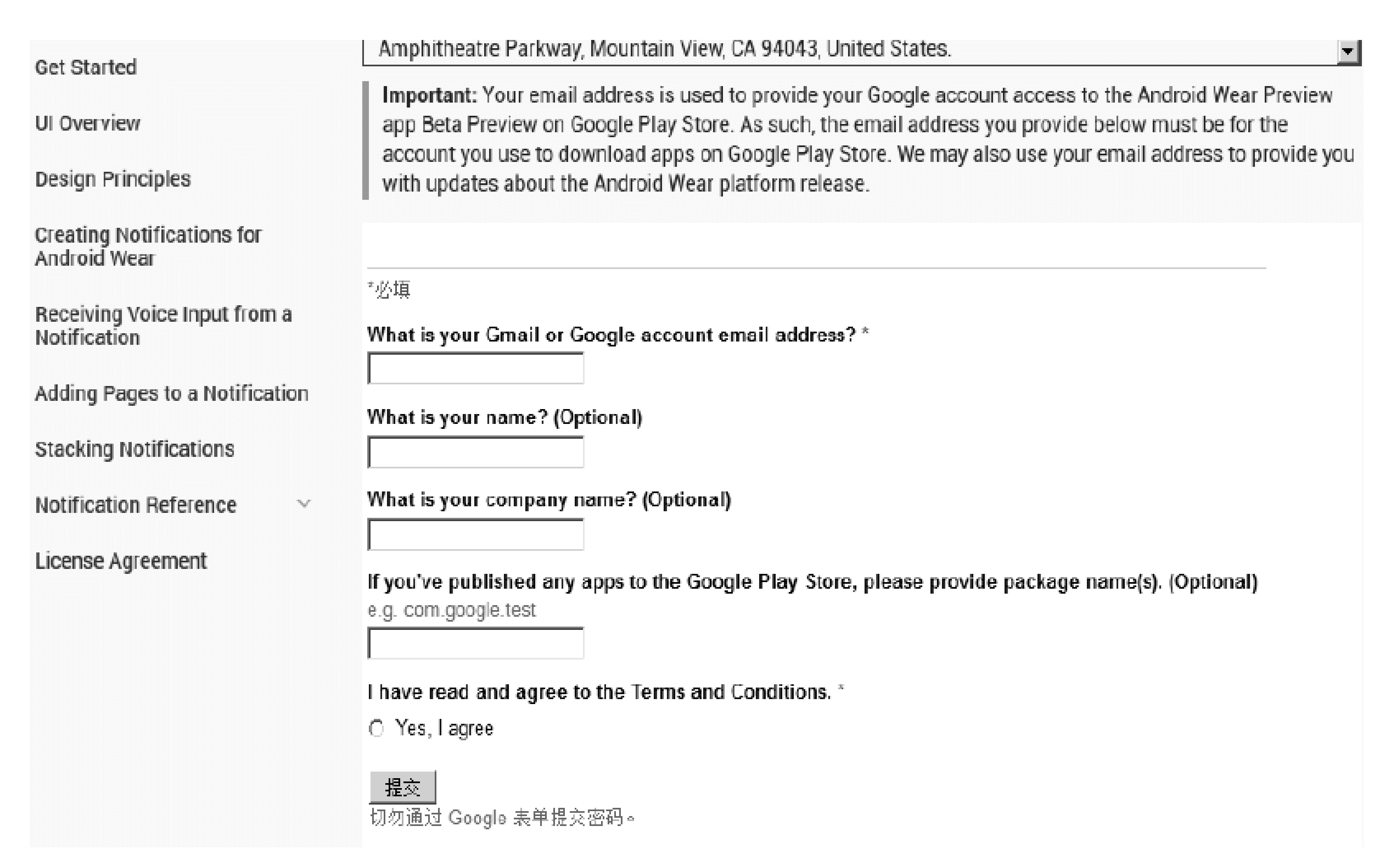


图 8-10　注册界面

（3）输入 Gmail 账户信息后单击“提交”按钮，等待谷歌发送回复的邮件信息，如图 8-11 所示。

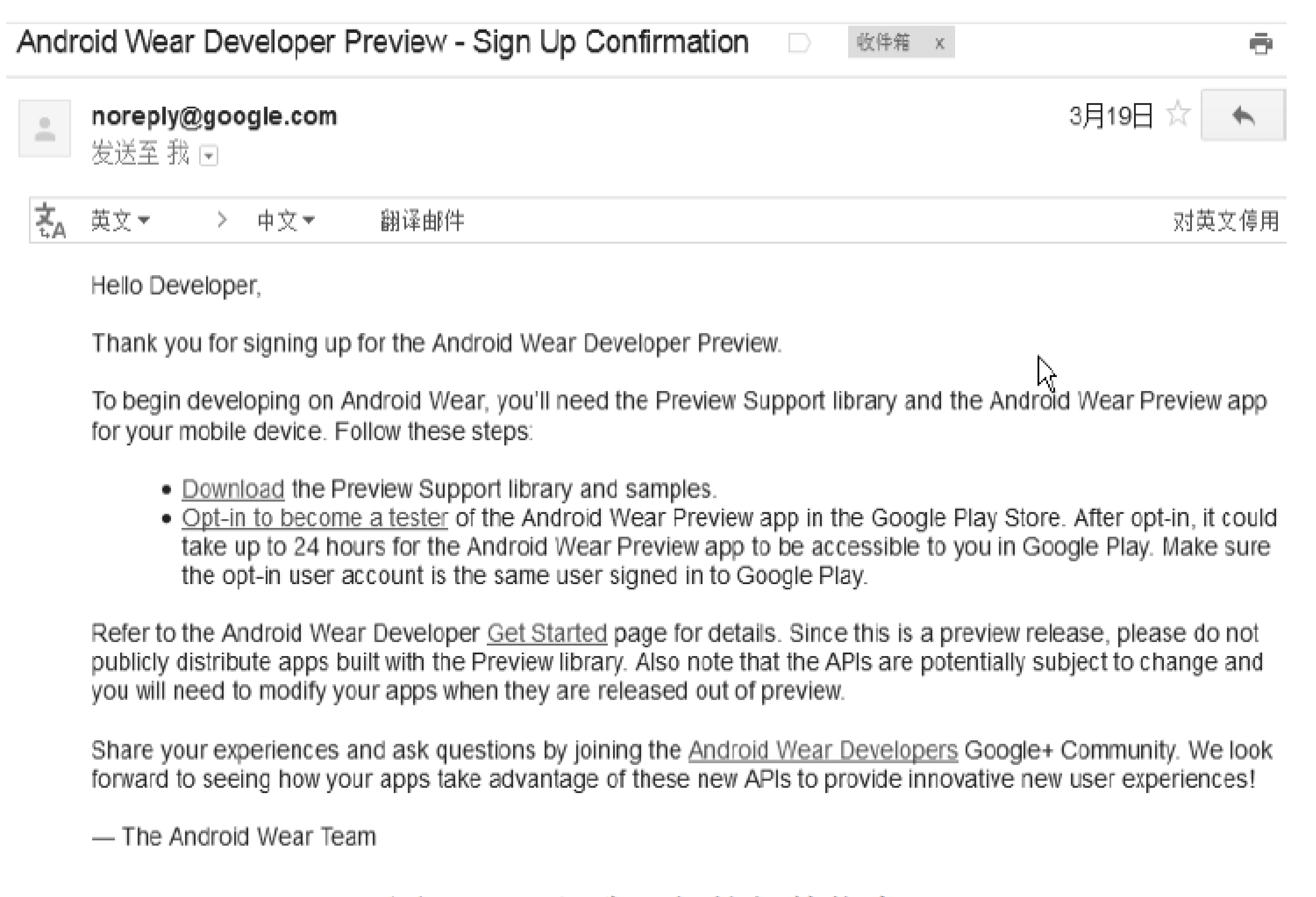


图 8-11　谷歌回复的邮件信息

通过邮件中的链接可以下载 Android Wear 预览版的开发程序包和演示实例，下载后的压缩包是 AndroidWearPreview.zip，解压缩后的效果如图 8-12 所示。

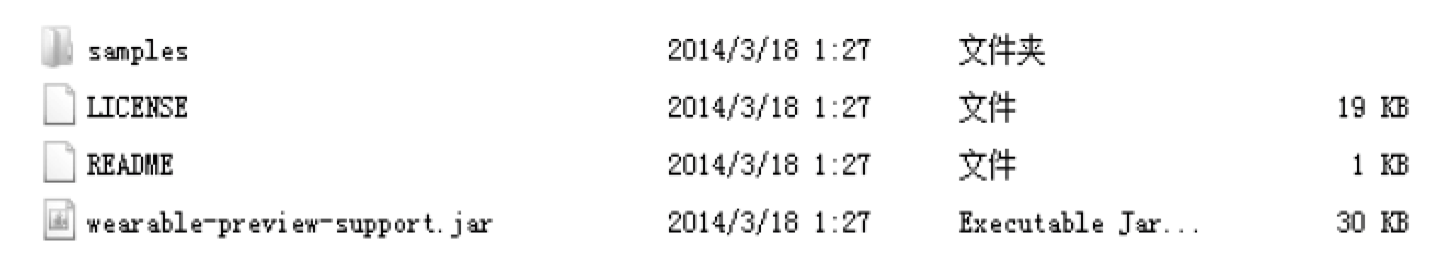
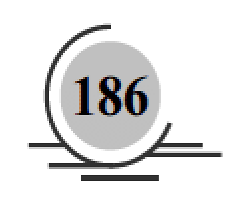


图 8-12　解压缩 AndroidWearPreview.zip



（4）检查 Android SDK 工具的版本 22.6 或更高，如果当前 Android SDK 工具的版本低于 22.6，则必须进行更新。

（5）在图 8-13 所示的界面中创建一个 Android 模拟器，Android Wear 要求的最低版本是 Android 4.4.2，选择“armeabi-v7a”类型。

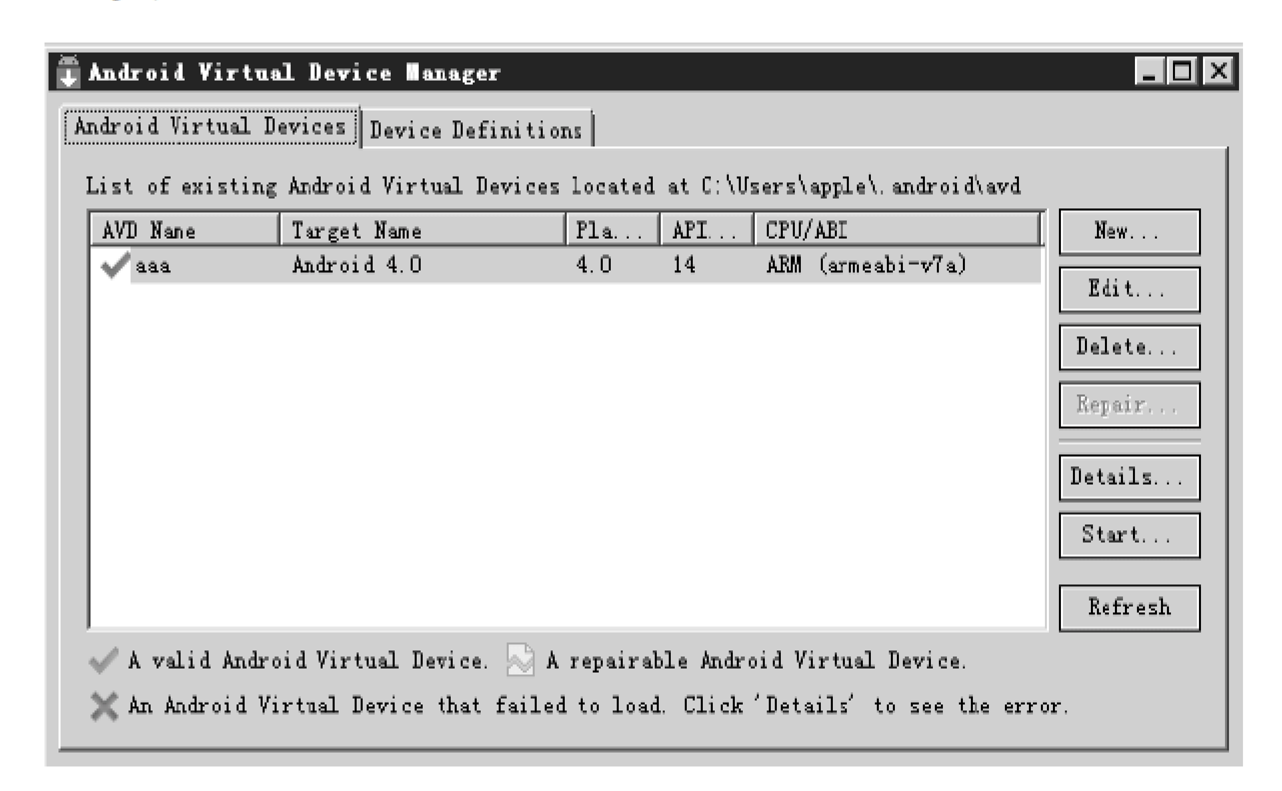


图 8-13　准备创建一个 Android 模拟器

（6）单击图 8-13 中的 New 按钮，在新界面中进行如下设置。

- name：设置创建模拟器的名字为wear。
- Target：设置此值最低为Android 4.4.2 - API Level 19。
- CPU/ABI：设置此值为Android Wear ARM（armeabi-v7a）。
- Skin：用于设置Android Wear 的外观，现在Android Wear只有两种外观，分别是方形AndroidWearSquare或圆形AndroidWearRound。
- 其他选项：设置为默认值即可。

设置后的界面效果如图 8-14 所示。

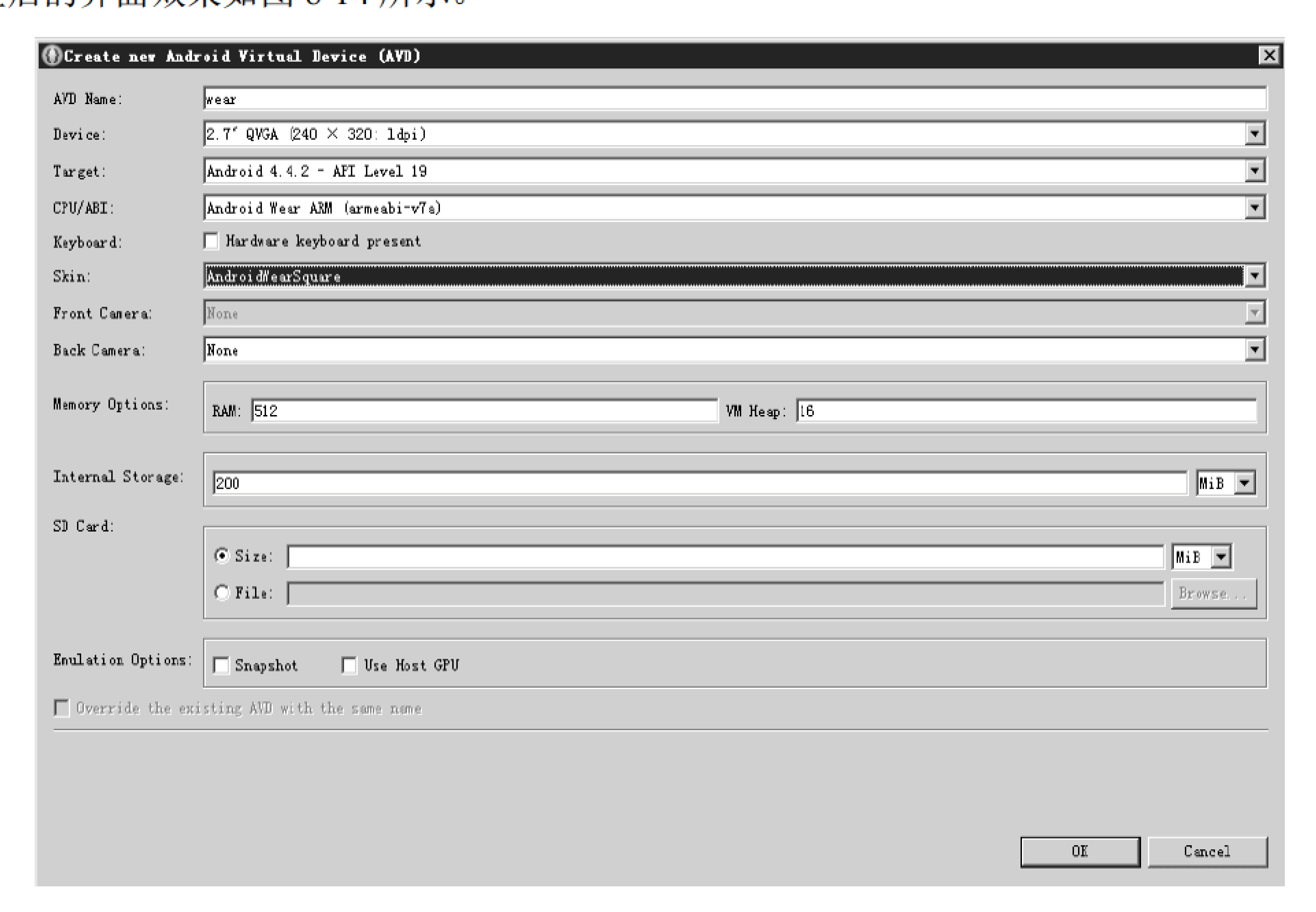


图 8-14　创建了一个方形 Android Wear 模拟器

单击 OK 按钮完成创建，单击 Start 按钮可以运行这个模拟器，运行后的效果如图 8-15 所示。

另外，通过谷歌回复的 Gmail 邮件可知，可以登录 https://play.google.com/apps/testing/com.google. android.wearablepreview.app 下载 Android Wear Preview，当然最简单的方法是从 Paly 商店下载获取，如图 8-16 所示。

另外，也可以登录 https://plus.google.com/communities/11338122747302 1565406 进入测试人员社区，在这里可以和一线开发人员进行交流，如图 8-17 所示。



图 8-15　模拟器运行效果

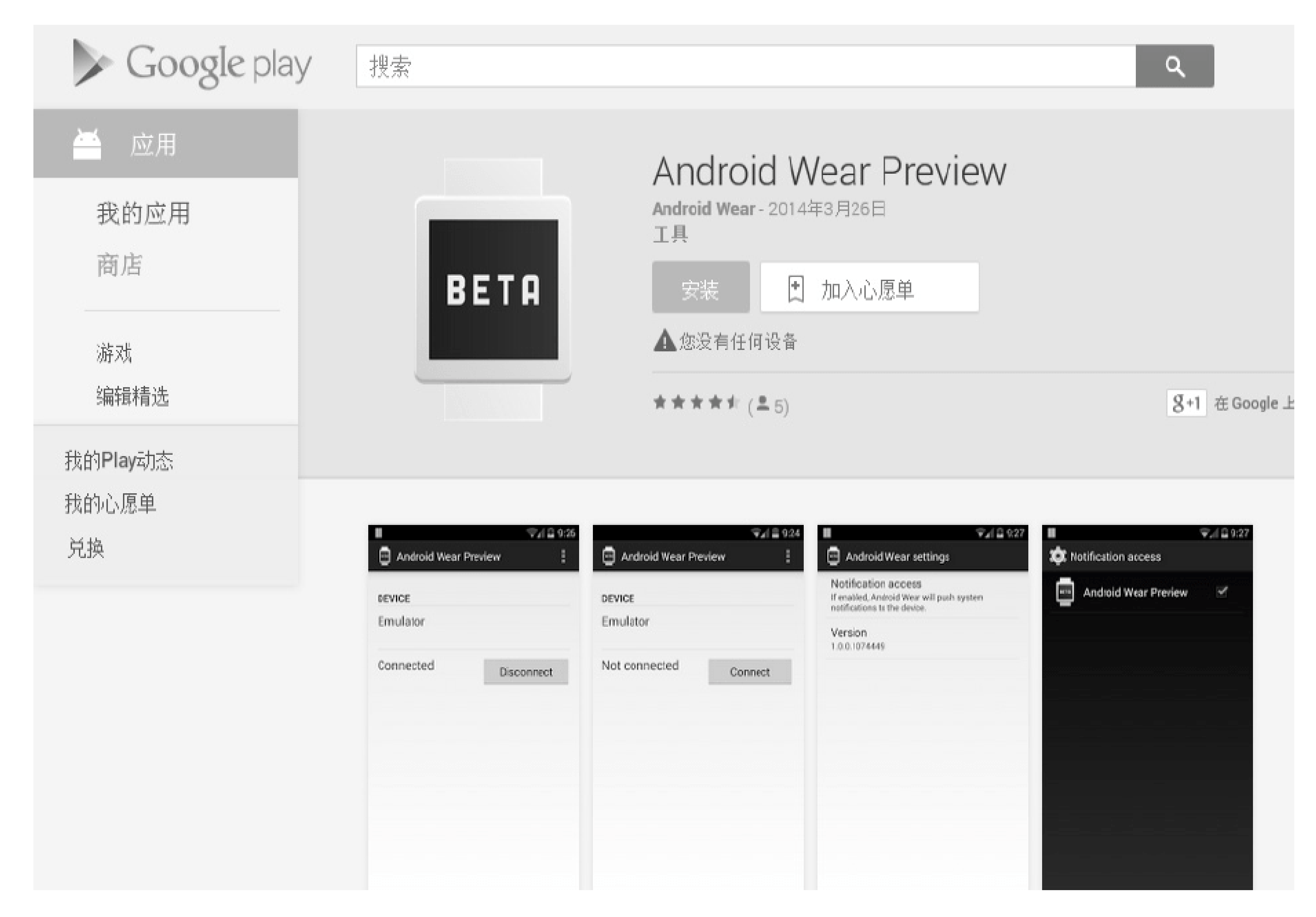


图 8-16　从 Paly 商店下载获取 Android Wear Preview

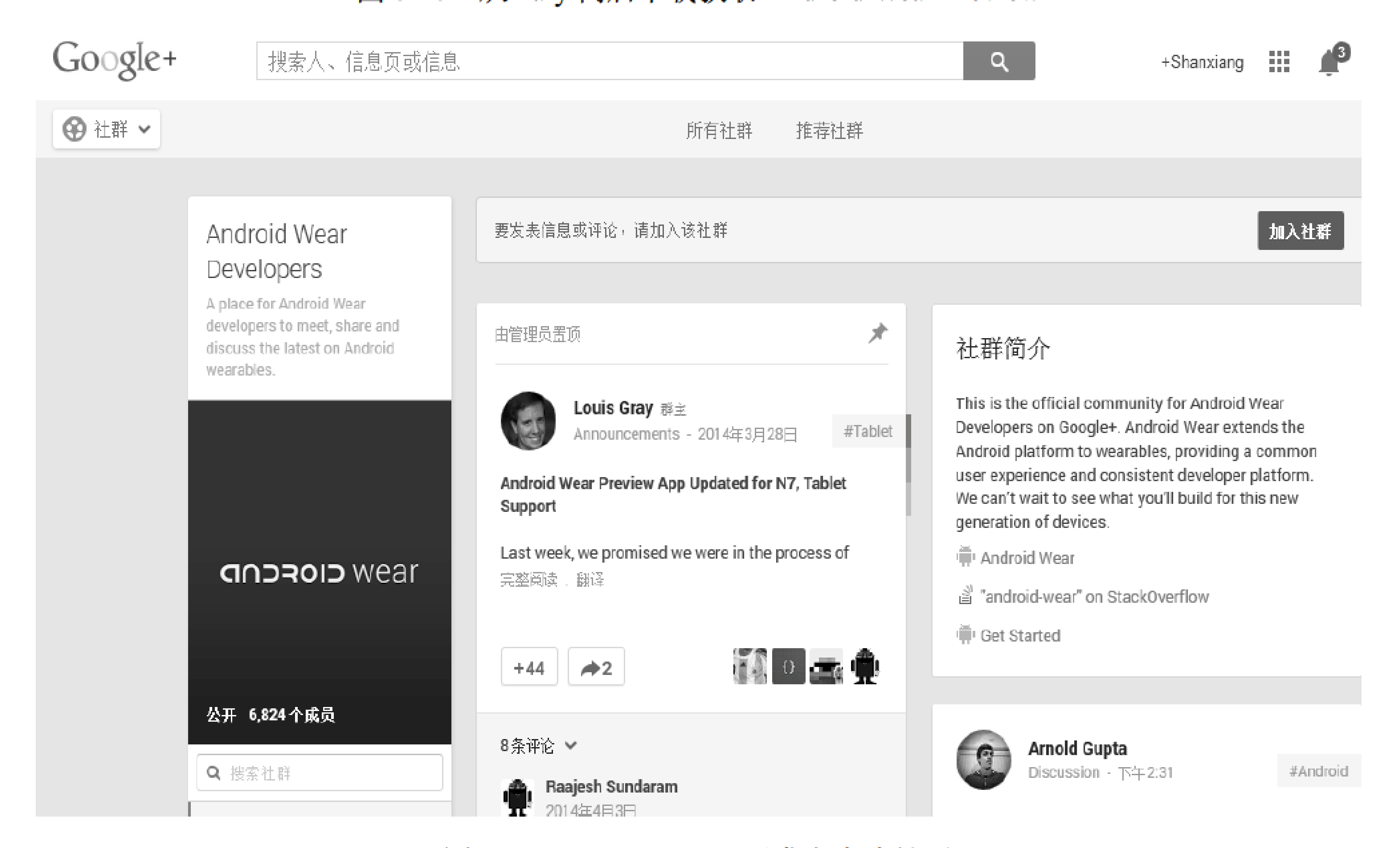
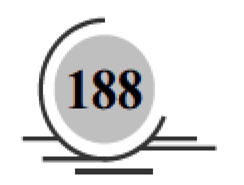


图 8-17　Android Wear 开发者交流社区



8.3　开发 Android Wear 程序

知识点讲解：光盘:视频\知识点\第 8 章\开发 Android Wear 程序.avi

在搭建完 Android Wear 开发环境之后，接下来开始讲解开发 Android Wear 程序的基本知识。在本节将首先讲解开发 Android Wear 程序的知识，然后通过一个演示实例来讲解具体开发过程。

8.3.1　创建通知

当一个手机或平板电脑等 Android 设备连接到一个 Android Wear 时，所有的通知在设备之间都是共享的。在 Android Wear 中，每个通知都会以新卡片背景流的样式出现，如图 8-18 所示。

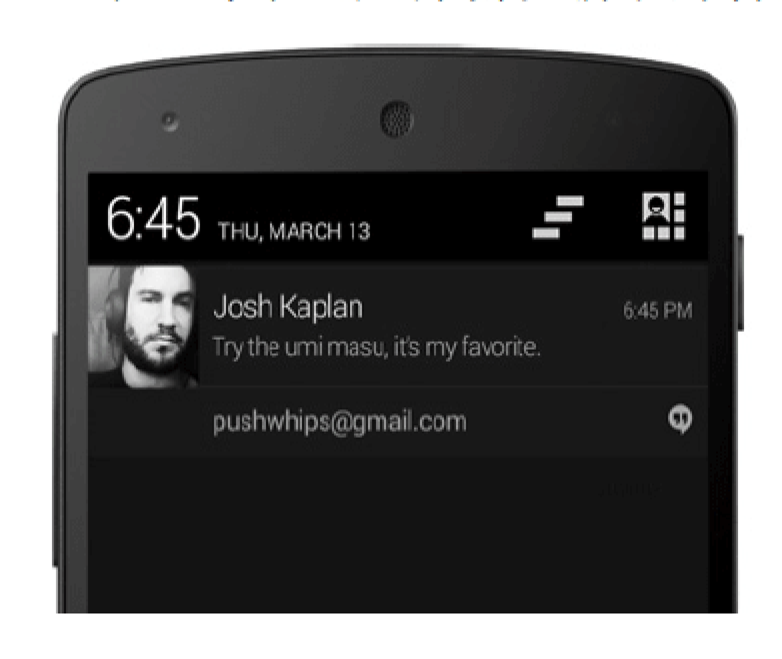




图 8-18　出现通知

由此可见，无须经过多少工作量，便可以在 Android Wear 设备中创建一个通知应用程序用。但是为了提高用户体验，当用户面对一个通知时，再通过声音来回复。

（1）引入需要的类

在开发 Android Wear 应用程序之前，必须首先详细阅读开发者预览文档。在该文档文件中提到，Android Wear 应用程序必须包括 V4 支持库和开发者预览版支持库。所以开始的时候，应该在你的项目代码中包括下面的引入文件：

```
import android.preview.support.wearable.notifications.*;
import android.preview.support.v4.app.NotificationManagerCompat;
import android.support.v4.app.NotificationCompat;
```

（2）通过提醒 Builder 创建通知

在 Android Wear 中，通过用 V4 支持库可以实现最新的通知等功能，例如用操作按钮和大图标创建通知。例如在下面的演示代码中，使用 NotificationCompat API 结合新的 NotificationManagerCompat API 可以创建并发布通知。

```
int notificationId = 001;
// Build intent for notification content
Intent viewIntent = new Intent(this, ViewEventActivity.class);
viewIntent.putExtra(EXTRA_EVENT_ID, eventId);
PendingIntent viewPendingIntent =
        PendingIntent.getActivity(this, 0, viewIntent, 0);

NotificationCompat.Builder notificationBuilder =
```

```
            new NotificationCompat.Builder(this)
            .setSmallIcon(R.drawable.ic_event)
            .setContentTitle(eventTitle)
            .setContentText(eventLocation)
            .setContentIntent(viewPendingIntent);

// Get an instance of the NotificationManager service
NotificationManagerCompat notificationManager =
            NotificationManagerCompat.from(this);

// Build the notification and issues it with notification manager.
notificationManager.notify(notificationId, notificationBuilder.build());
```

通过上述代码，当上述通知出现在手持设备中时，用户可以调用指定的 setcontentintent()方法通过触摸的方式通知 PendingIntent。当这个通知出现在 Android Wear 中时，用户也可以用通知操作来调用在手持设备上的意图。

（3）添加动作按钮

除了通过 setcontentintent()定义的主要操作外，还可以通过传递 PendingIntent 到 addaction()方法的方式添加其他操作。例如下面的代码显示了和前面类型相同的通知，但是增加了一个在地图上实现定位的事件操作。

```
// Build an intent for an action to view a map
Intent mapIntent = new Intent(Intent.ACTION_VIEW);
Uri geoUri = Uri.parse("geo:0,0?q=" + Uri.encode(location));
mapIntent.setData(geoUri);
PendingIntent mapPendingIntent =
            PendingIntent.getActivity(this, 0, mapIntent, 0);

NotificationCompat.Builder notificationBuilder =
            new NotificationCompat.Builder(this)
            .setSmallIcon(R.drawable.ic_event)
            .setContentTitle(eventTitle)
            .setContentText(eventLocation)
            .setContentIntent(viewPendingIntent)
            .addAction(R.drawable.ic_map,
                        getString(R.string.map), mapPendingIntent);
```

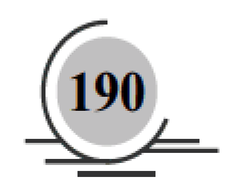
（4）为通知添加一个大视图

在手持设备上，用户可以通过扩大通知卡片的方式来查看通知内容。在 Android Wear 设备中，大视图的内容是默认可见的。当在通知中添加扩展的内容后，可以调用 NotificationCompat.Builder 对象中的 setStyle()方法实现 bigtextstyle 或 inboxstyle 样式实例。

例如在下面的代码中，添加了 NotificationCompat.bigtextstyle 实例的事件通知，这样可以包括完整的事件描述，包括可以提供比 setcontenttext()空间更多的文本内容。

```
BigTextStyle bigStyle = new NotificationCompat.BigTextStyle();
bigStyle.bigText(eventDescription);

NotificationCompat.Builder notificationBuilder =
            new NotificationCompat.Builder(this)
            .setSmallIcon(R.drawable.ic_event)
            .setLargeIcon(BitmapFractory.decodeResource(
```



```
                getResources(), R.drawable.notif_background))
        .setContentTitle(eventTitle)
        .setContentText(eventLocation)
        .setContentIntent(viewPendingIntent)
        .addAction(R.drawable.ic_map,
                getString(R.string.map), mapPendingIntent)
        .setStyle(bigStyle);
```

注意：可以使用setlargeicon()方法为任何通知添加一个背景图像。

（5）为设备添加新的功能

在 Android Wear 预览版的支持库中提供了很多新的 API，通过这些 API 可以在穿戴设备中提高通知用户体验。例如可以添加额外的页面的内容，或添加用户使用语音输入文本的响应功能。通过使用这些新的 API，通过实例的 NotificationCompat.Builder()构造函数可以添加新的功能。例如下面的演示代码。

```
// Create a NotificationCompat.Builder for standard notification features
NotificationCompat.Builder notificationBuilder =
        new NotificationCompat.Builder(mContext)
        .setContentTitle("New mail from " + sender.toString())
        .setContentText(subject)
        .setSmallIcon(R.drawable.new_mail);

// Create a WearablesNotification.Builder to add special functionality for wearables
Notification notification =
        new WearableNotifications.Builder(notificationBuilder)
        .setHintHideIcon(true)
        .build();
```

在上述代码中，方法 setHintHideIcon()的功能是从通知卡中移除应用程序图标。方法 setHintHideIcon()是一个新的通知功能，可以从 WearableNotifications.Builder 对象中生成。

当想要推送传递你的通知时，一定要始终使用 NotificationManagerCompat API，例如下面的演示代码。

```
NotificationManagerCompat notificationManager =
        NotificationManagerCompat.from(this);

// Build the notification and issues it with notification manager.
notificationManager.notify(notificationId, notification);
```

注意：在笔者写作此书时，Android Wear开发者预览版API只是为了开发和测试而推出的，并不是为了编写出具体的应用程序。谷歌在正式公布Android Wear SDK之前，上述开发流程只是待定的。

8.3.2　创建声音

如果在创建的通知中包含了文本回复功能，例如回复一封邮件，在通常情况下会在手持设备上启动一个 Activity。当我们的通知显示在穿戴设备上时，可以允许用户使用语音输入口诉一个回复，还可以提供预先设置的文本信息让用户选择。当用户使用语音回复或者选择预设信息时，系统会发送信息到与手持设备相连的应用，该信息以一个附加品的形式与我们定义使用的通知行动的 Intent 相关联，如图 8-19 所示。



图 8-19　声音回复

注意：在安卓模拟器上开发时，即使在语音输入域，也必须要使用文本回复，所以需确保在AVD设置上已激活了Hardware keyboard present。

（1）定义远程回复

在 Android Wear 中创建支持语音输入的行动时，首先需要使用 RemoteInput.Builder APIs 创建一个 RemoteInput 的实例。RemoteInput.Builder 构造器获取一个 String 类型的值，系统会将这个值作为一个 key 传递给 Intent extra，这个 Intent 可以将回复信息传送到手持设备中的应用程序。例如在下面的代码中创建了一个新的 RemoteInput 对象，功能是提供自定义标签给语音输入命令。

```
// 传送给行动 intent 的 key 的字符串
private static final String EXTRA_VOICE_REPLY = "extra_voice_reply";
String replyLabel = getResources().getString(R.string.reply_label);
RemoteInput remoteInput = new RemoteInput.Builder(EXTRA_VOICE_REPLY)
        .setLabel(replyLabel)
        .build();
```

（2）添加预置文本进行回复

除了支持语音输入外，在 Android Wear 中还可以提供最多 5 条预置文本回复信息以供用户进行快速回复。实现方法是调用 setChoices()方法，并将字符串数组传递给它。例如可以在资源数组中定义如下所示的回复。

```
res/values/strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="reply_choices">
        <item>Yes</item>
        <item>No</item>
        <item>Maybe</item>
    </string-array>
</resources>
```

效果如图 8-20 所示。

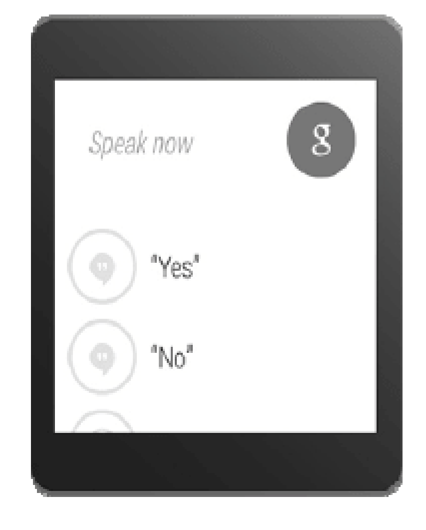
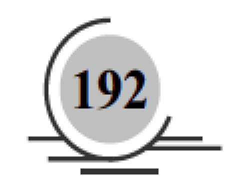


图 8-20　添加的回复数值



然后通过如下代码释放 String 数组并将其添加到 RemoteInput 中。

```
String replyLabel = getResources().getString(R.string.reply_label);
String[ ] replyChoices = getResources().getStringArray(R.array.reply_choices);

RemoteInput remoteInput = new RemoteInput.Builder(EXTRA_VOICE_REPLY)
        .setLabel(replyLabel)
        .setChoices(replyChoices)
        .build();
```

（3）为主行动接收语音输入

在 Android Wear 应用中，如果 Reply 是我们应用程序的主行动（由 setContentIntent() 方法定义），那么需要使用 addRemoteInputForContentIntent()方法将 RemoteInput 添加到主行动上。例如下面的演示代码。

```
// 为回复行动创建 intent
Intent replyIntent = new Intent(this, ReplyActivity.class);
PendingIntent replyPendingIntent =
        PendingIntent.getActivity(this, 0, replyIntent, 0);
// 创建通知
NotificationCompat.Builder replyNotificationBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.ic_new_message)
        .setContentTitle("Message from Travis")
        .setContentText("I love key lime pie!")
        .setContentIntent(replyPendingIntent);
//创建运程回复<语音>
RemoteInput remoteInput = new RemoteInput.Builder(EXTRA_VOICE_REPLY)
        .setLabel(replyLabel)
        .build();
//创建穿戴设备的通知并添加语音输入
Notification replyNotification =
        new WearableNotifications.Builder(replyNotificationBuilder)
        .addRemoteInputForContentIntent(remoteInput)
        .build();
```

通过使用 addRemoteInputForContentIntent()方法，将 RemoteInput 对象添加到通知的主行动中后，通常 Open 按钮会显示为 Reply 按钮，当用户在 Android Wear 上选择它时，就会启动语音输入 UI 视图界面。

（4）为次行动设置语音输入

如果 Reply 动作不是我们创建通知的主动作，而只是为次行动激活语音输入，那么可以添加 RemoteInput 到新的行动按钮（由 Action 对象定义）。通过 Action.Builder()构造器实例化 Action，它会给行动按钮添加一个 icon 和文本标签，加上 PendingIntent，当用户选择这个行动时，系统会使用它调用我们的应用。例如下面的演示代码。

```
// 创建一个 pending intent，当用户选择这个行动时，会启用这个 intent
Intent replyIntent = new Intent(this, ReplyActivity.class);
PendingIntent pendingReplyIntent =
        PendingIntent.getActivity(this, 0, replyIntent, 0);

//创建远程输入 Create the remote input
RemoteInput remoteInput = new RemoteInput.Builder(EXTRA_VOICE_REPLY)
```

```
            .setLabel(replyLabel)
            .build();

//创建通知行动  Create the notification action
Action replyAction = new Action.Builder(R.drawable.ic_message,
            "Reply", pendingIntent)
            .addRemoteInput(remoteInput)
            .build();
```

然后为 Action 添加 RemoteInput.Builder，使用 addAction()方法为 WearableNotifications.Builder 添加 Action。例如下面的演示代码。

```
//创建基本的通知创建者  Create basic notification builder
NotificationCompat.Builder replyNotificationBuilder =
            new NotificationCompat.Builder(this)
            .setContentTitle("New message");

// 创建通知行动并添加远程输入 Create the notification action and add remote input
Action replyAction = new Action.Builder(R.drawable.ic_message,
            "Reply", pendingIntent)
            .addRemoteInput(remoteInput)
            .build();

// 创建穿戴设备的通知并添加行动 Create wearable notification and add action
Notification replyNotification =
            new WearableNotifications.Builder(replyNotificationBuilder)
            .addAction(replyAction)
            .build();
```

现在，当用户在 Android Wear 设备上选择 Reply 时，系统提示用户使用语音输入（如果提供了预置回复，则会显示预置列表）。当用户完成回复时，系统会调用与该行动关联的 intent，并添加作为字符值的 EXTRA_VOICE_REPLY extra（传递给 RemoteInput.Builder 构造器的字符串）到用户信息。

8.3.3 给通知添加页面

当想为 Android Wear 设备提供更多的信息，而且不需要用户使用手持设备打开应用时，可以在 Android Wear 上为通知添加一个或若干页面，附加的页面会在主通知卡片的右边立即显示出来，如图 8-21 所示。

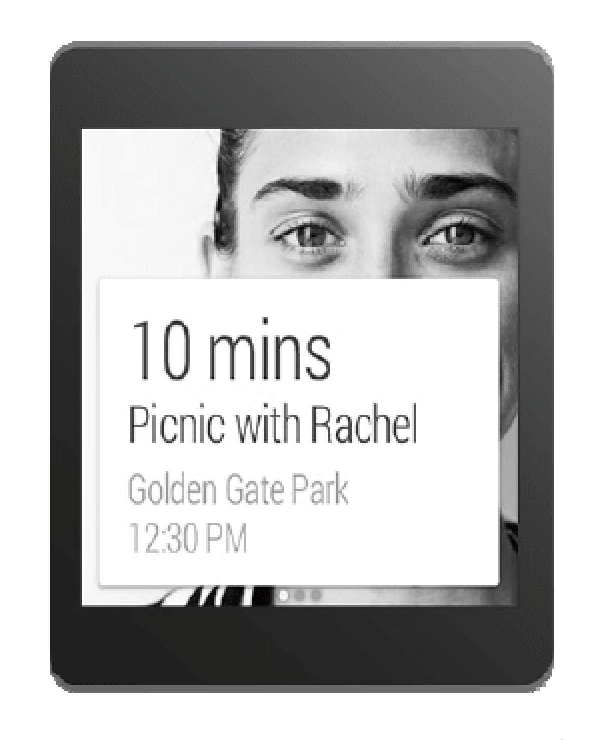
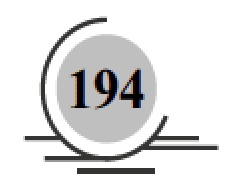




图 8-21　给通知添加页面

当创建多张页面时，第一步需要把通知显示在手机或平板设备，也就是先创建一个主通知（第一张页面），然后使用 addPage()方法每次添加一张页面，或者使用 addPages()方法从 Collection 对象添加



若干页面。例如下面的演示代码。

```
//为主通知创建 Builder
NotificationCompat.Builder notificationBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.new_message)
        .setContentTitle("Page 1")
        .setContentText("Short message")
        .setContentIntent(viewPendingIntent);

//为第二张页面创建 big text 风格
BigTextStyle secondPageStyle = new NotificationCompat.BigTextStyle();
secondPageStyle.setBigContentTitle("Page 2")
                .bigText("A lot of text...");

// Create second page notification
Notification secondPageNotification =
        new NotificationCompat.Builder(this)
        .setStyle(secondPageStyle)
        .build();

// Create main notification and add the second page
Notification twoPageNotification =
        new WearableNotifications.Builder(notificationBuilder)
        .addPage(secondPageNotification)
        .build();
```

8.3.4　通知堆

当为手持设备创建通知时，大家可能习惯于把同类型的通知放在一个汇总通知里。例如，如果你的应用创建了接收信息的通知，当接收到多条信息时不会显示多条通知在手持设备上，而是使用单条通知。此时只需要提供汇总信息就可以了，例如“两条新信息”的提示。但是在 Android Wear 设备上，汇总信息没什么作用，因为用户无法在 Android Wear 设备上逐条阅读细节（因为他们必须在手持设备上打开应用查看更多信息）。为了 Android Wear 设备，需要把所有的通知聚集到一个堆里。通知堆以单张卡片的形式存在，用户可以展开它逐条查看，新的 setGroup()方法为你提供了可能，虽然在手持设备上，它仍然仅提供一条汇总信息，如图 8-22 所示。

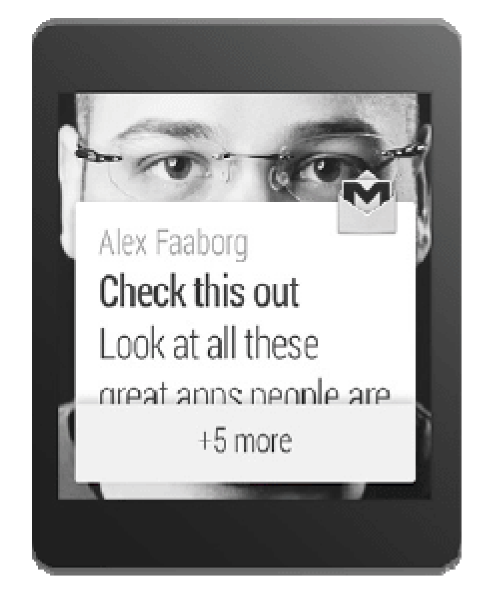
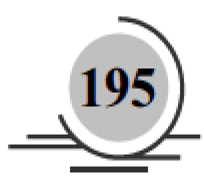




图 8-22　通知堆演示界面

（1）将通知逐条添加到 Group

为了在 Android Wear 中创建堆，需要为每条通知调用 setGroup()方法，并将唯一的 group key 传递



给它们。例如下面的演示代码。

```
final static String GROUP_KEY_EMAILS = "group_key_emails";

NotificationCompat.Builder builder = new NotificationCompat.Builder(mContext)
        .setContentTitle("New mail from " + sender)
        .setContentText(subject)
        .setSmallIcon(R.drawable.new_mail);

Notification notif = new WearableNotifications.Builder(builder)
        .setGroup(GROUP_KEY_EMAILS)
        .build();
```

在默认情况下，会以我们的添加顺序来展现通知，最新的通知显示在头部。但是也可以通过传递一个位置值给 setGroup()的第二个参数，在 group 里定义一个特殊的位置。

（2）添加一个汇总通知

在 Android Wear 应用中，提供一个汇总通知给手持设备是很重要的。除了逐条添加通知到同一个通知堆外，建议仍添加一个汇总通知，不过设置它的次序为 GROUP_ORDER_SUMMARY。例如下面的演示代码。

```
Notification summaryNotification = new WearableNotifications.Builder(builder)
        .setGroup(GROUP_KEY_EMAILS, WearableNotifications.GROUP_ORDER_SUMMARY)
        .build();
```

这条通知不会显示在 Android Wear 设备上的通知堆里，只会显示在手持设备上的唯一通知上。

8.3.5 通知语法介绍

（1）android.preview.support.v4.app

用 NotificationCompat.Builder 对象来设定通知的 UI 信息和行为，调用 NotificationCompat.Builder.build()来创建通知，调用 NotificationManager.notify()来发送通知。

一条通知必须包含如下信息：

- 一个小图标：用setSmallIcon()来设置。
- 一个标题：用setContentTitle()来设置。
- 详情文字：用setContentText()来设置。

（2）android.preview.support.wearable.notifications 表示可穿戴设备的通知信息类型，其中不同的输入类型信息如表 8-1 所示。

表 8-1 通知信息类型表

类　型	说　明
RemoteInput	远程输入类，可穿戴设备输入
RemoteInput.Builder	生成 RemoteInput 的目标
WearableNotifications	可穿戴设备类型的通知
WearableNotifications.Action	可穿戴设备类型通知的行为动作
WearableNotifications.Action.Builder	生成类 WearableNotifications.Action 对象
WearableNotifications.Builder	一个 NotificationCompat.Builder 生成器对象，为可穿戴的扩展功能提供通知方法

例如在下面的代码中，通过注释详细讲解并演示了各个 Android Wear 对象的基本用法。

```
 int notificationId = 001;          //通知 id
 Intent replyIntent = new Intent(this, ReplyActivity.class); // 响应 Action，可以启动 Activity、Service 或者
Broadcast
  PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, replyIntent, 0);
  RemoteInput remoteInput = new RemoteInput.Builder("key")//响应输入，key 为返回 Intent 的 Extra 的 Key 值
       .setLabel("Select")       //输入页标题
       .setChoices(String[ ])     //输入可选项
       .build();
 Action replyAction = new Action.Builder(R.drawable, //WearableNotifications.Action.Builder 对应可穿戴设备的
Action 类
                               "Reply", pendingIntent) //对应 pendingIntent
                               .addRemoteInput(remoteInput)
                               .build();

 NotificationCompat.Builder notificationBuilder =  new NotificationCompat.Builder(mContext) //标准通知创建
               .setContentTitle(title).setContentText(subject).setSmallIcon(R.drawable).setStyle(style)
  .setLargeIcon(bitmap)                           //设置可穿戴设备显示的背景图
  .setContentIntent(pendingIntent) //可穿戴设备左滑，有默认 Open 操作，对应手机端的点击通知
  .addAction(R.drawable, String, pendingIntent);    //增加一个操作，可加多个
 Notification notification = new WearableNotifications.Builder(notificationBuilder)    //创建可穿戴类通知，为通知
增加可穿戴设备新特性，必须与兼容包里的 NotificationManager 对应，否则无效
               .setHintHideIcon(true)           //隐藏应用图标
             .addPages(notificationPages)       //增加 Notification 页
             .addAction(replyAction)            //对应上页，pendingIntent 可操作项
 .addRemoteInputForContentIntent(replyAction) //可为 ContentIntent 替换默认的 Open 操作
 .setGroup(GROUP_KEY, WearableNotifications.GROUP_ORDER_SUMMARY) //为通知分组
 .setLocalOnly(true)                             //可设置只在本地显示
 .setMinPriority()                               //设置只在可穿戴设备上显示通知
             .build();
 NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);//获得 Manager
 notificationManager.notify(notificationId, notificationBuilder.build());//发送通知
```

8.4　实战演练——开发一个 Android Wear 程序

知识点讲解：光盘:视频\知识点\第 8 章\开发一个 Android Wear 程序.avi

在接下来的内容中，将通过一个具体实例来讲解开发一个 Android Wear 程序的方法。

实例	功能	源码路径
实例 8-1	开发一个 Android Wear 程序	光盘:\daima\27\wearmaster

本实例的具体实现流程如下。

（1）编写布局文件 activity_main.xml，具体实现代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.ezhuk.wear.MainActivity">
    <TextView
        android:text="@string/hello_world"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```

（2）编写值文件 strings.xml，功能是设置通知的文本内容，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Wear</string>
    <string name="hello_world">Test</string>
    <string name="content_title">Basic Notification</string>
    <string name="content_text">Sample text.</string>
    <string name="page1_title">Page 1</string>
    <string name="page1_text">Sample text 1.</string>
    <string name="page2_title">Page 2</string>
    <string name="page2_text">Sample text 2.</string>
    <string name="action_title">Action Title</string>
    <string name="action_text">Action text.</string>
    <string name="action_button">Action</string>
    <string name="action_label">Action</string>
    <string name="summary_title">Summary Title</string>
    <string name="summary_text">Summary text.</string>
    <string-array name="input_choices">
        <item>First item</item>
        <item>Second item</item>
        <item>Third item</item>
    </string-array>
</resources>
```

（3）编写文件 MainActivity.java 实现程序的主 Activity，功能是载入 Android Wear 的通知类 NotificationUtils，调用不同的 showNotificationXX 方法显示通知信息，具体实现代码如下所示。

```
package com.ezhuk.wear;

import android.app.Activity;
import android.os.Bundle;

import static com.ezhuk.wear.NotificationUtils.*;


public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

```
    @Override
    protected void onResume() {
        super.onResume();

        showNotification(this);
        showNotificationNoIcon(this);
        showNotificationMinPriority(this);
        showNotificationBigTextStyle(this);
        showNotificationBigPictureStyle(this);
        showNotificationInboxStyle(this);
        showNotificationWithPages(this);
        showNotificationWithAction(this);
        showNotificationWithInputForPrimaryAction(this);
        showNotificationWithInputForSecondaryAction(this);
        showGroupNotifications(this);
    }

    @Override
    protected void onPause() {
        super.onPause();
    }
}
```

（4）编写文件 NotificationUtils.java，功能是定义各种不同类型 showNotificationXX 的通知方法，具体实现代码如下所示。

```
import android.app.Notification;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.preview.support.v4.app.NotificationManagerCompat;
import android.preview.support.wearable.notifications.RemoteInput;
import android.preview.support.wearable.notifications.WearableNotifications;
import android.support.v4.app.NotificationCompat;


public class NotificationUtils {
    private static final String ACTION_TEST = "com.ezhuk.wear.ACTION";
    private static final String ACTION_EXTRA = "action";

    private static final String NOTIFICATION_GROUP = "notification_group";

    public static void showNotification(Context context) {
        NotificationCompat.Builder builder =
                new NotificationCompat.Builder(context)
                        .setSmallIcon(R.drawable.ic_launcher)
                        .setContentTitle(context.getString(R.string.content_title))
                        .setContentText(context.getString(R.string.content_text));

        NotificationManagerCompat.from(context).notify(0,
                new WearableNotifications.Builder(builder)
```

```
                        .build());
    }

    public static void showNotificationNoIcon(Context context) {
        NotificationCompat.Builder builder =
                new NotificationCompat.Builder(context)
                        .setSmallIcon(R.drawable.ic_launcher)
                        .setContentTitle(context.getString(R.string.content_title))
                        .setContentText(context.getString(R.string.content_text));

        NotificationManagerCompat.from(context).notify(1,
                new WearableNotifications.Builder(builder)
                        .setHintHideIcon(true)
                        .build());
    }

    public static void showNotificationMinPriority(Context context) {
        NotificationCompat.Builder builder =
                new NotificationCompat.Builder(context)
                        .setSmallIcon(R.drawable.ic_launcher)
                        .setContentTitle(context.getString(R.string.content_title))
                        .setContentText(context.getString(R.string.content_text));

        NotificationManagerCompat.from(context).notify(2,
                new WearableNotifications.Builder(builder)
                        .setMinPriority()
                        .build());
    }

    public static void showNotificationWithStyle(Context context,
                                                 int id,
                                                 NotificationCompat.Style style) {
        Notification notification = new WearableNotifications.Builder(
                new NotificationCompat.Builder(context)
                        .setSmallIcon(R.drawable.ic_launcher)
                        .setStyle(style))
                .build();

        NotificationManagerCompat.from(context).notify(id, notification);
    }

    public static void showNotificationBigTextStyle(Context context) {
        showNotificationWithStyle(context, 3,
                new NotificationCompat.BigTextStyle()
                        .setSummaryText(context.getString(R.string.summary_text))
                        .setBigContentTitle("Big Text Style")
                        .bigText("Sample big text."));
    }

    public static void showNotificationBigPictureStyle(Context context) {
        showNotificationWithStyle(context, 4,
                new NotificationCompat.BigPictureStyle()
                        .setSummaryText(context.getString(R.string.summary_text))
```

```
                    .setBigContentTitle("Big Picture Style")
                    .bigPicture(BitmapFactory.decodeResource(
                            context.getResources(), R.drawable.background)));
}

public static void showNotificationInboxStyle(Context context) {
    showNotificationWithStyle(context, 5,
            new NotificationCompat.InboxStyle()
                    .setSummaryText(context.getString(R.string.summary_text))
                    .setBigContentTitle("Inbox Style")
                    .addLine("Line 1")
                    .addLine("Line 2"));
}

public static void showNotificationWithPages(Context context) {
    NotificationCompat.Builder builder =
            new NotificationCompat.Builder(context)
                    .setSmallIcon(R.drawable.ic_launcher)
                    .setContentTitle(context.getString(R.string.page1_title))
                    .setContentText(context.getString(R.string.page1_text));

    Notification second = new NotificationCompat.Builder(context)
            .setSmallIcon(R.drawable.ic_launcher)
            .setContentTitle(context.getString(R.string.page2_title))
            .setContentText(context.getString(R.string.page2_text))
            .build();

    NotificationManagerCompat.from(context).notify(6,
            new WearableNotifications.Builder(builder)
                    .addPage(second)
                    .build());
}

public static void showNotificationWithAction(Context context) {
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(Uri.parse(""));
    PendingIntent pendingIntent =
            PendingIntent.getActivity(context, 0, intent, 0);

    NotificationCompat.Builder builder =
            new NotificationCompat.Builder(context)
                    .setSmallIcon(R.drawable.ic_launcher)
                    .setContentTitle(context.getString(R.string.action_title))
                    .setContentText(context.getString(R.string.action_text))
                    .addAction(R.drawable.ic_launcher,
                            context.getString(R.string.action_button),
                            pendingIntent);

    NotificationManagerCompat.from(context).notify(7,
            new WearableNotifications.Builder(builder)
                    .build());
}
```

```
public static void showNotificationWithInputForPrimaryAction(Context context) {
    Intent intent = new Intent(ACTION_TEST);
    PendingIntent pendingIntent =
            PendingIntent.getActivity(context, 0, intent, 0);

    NotificationCompat.Builder builder =
            new NotificationCompat.Builder(context)
                    .setSmallIcon(R.drawable.ic_launcher)
                    .setContentTitle(context.getString(R.string.action_title))
                    .setContentText(context.getString(R.string.action_text))
                    .setContentIntent(pendingIntent);

    String[ ] choices =
            context.getResources().getStringArray(R.array.input_choices);

    RemoteInput remoteInput = new RemoteInput.Builder(ACTION_EXTRA)
            .setLabel(context.getString(R.string.action_label))
            .setChoices(choices)
            .build();

    NotificationManagerCompat.from(context).notify(8,
            new WearableNotifications.Builder(builder)
                    .addRemoteInputForContentIntent(remoteInput)
                    .build());
}

public static void showNotificationWithInputForSecondaryAction(Context context) {
    Intent intent = new Intent(ACTION_TEST);
    PendingIntent pendingIntent =
            PendingIntent.getActivity(context, 0, intent, 0);

    RemoteInput remoteInput = new RemoteInput.Builder(ACTION_EXTRA)
            .setLabel(context.getString(R.string.action_label))
            .build();

    WearableNotifications.Action action =
            new WearableNotifications.Action.Builder(
                    R.drawable.ic_launcher,
                    "Action",
                    pendingIntent)
            .addRemoteInput(remoteInput)
            .build();

    NotificationCompat.Builder builder =
            new NotificationCompat.Builder(context)
                    .setContentTitle(context.getString(R.string.action_title));

    NotificationManagerCompat.from(context).notify(9,
            new WearableNotifications.Builder(builder)
                    .addAction(action)
                    .build());
}
```

```
    public static void showGroupNotifications(Context context) {
        Notification first = new WearableNotifications.Builder(
                new NotificationCompat.Builder(context)
                        .setSmallIcon(R.drawable.ic_launcher)
                        .setContentTitle(context.getString(R.string.page1_title))
                        .setContentText(context.getString(R.string.page1_text)))
                .setGroup(NOTIFICATION_GROUP)
                .build();

        Notification second = new WearableNotifications.Builder(
                new NotificationCompat.Builder(context)
                        .setSmallIcon(R.drawable.ic_launcher)
                        .setContentTitle(context.getString(R.string.page2_title))
                        .setContentText(context.getString(R.string.page2_text)))
                .setGroup(NOTIFICATION_GROUP)
                .build();

        Notification summary = new WearableNotifications.Builder(
                new NotificationCompat.Builder(context)
                        .setSmallIcon(R.drawable.ic_launcher)
                        .setContentTitle(context.getString(R.string.summary_title))
                        .setContentText(context.getString(R.string.summary_text)))
                .setGroup(NOTIFICATION_GROUP, WearableNotifications.GROUP_ORDER_SUMMARY)
                .build();

        NotificationManagerCompat.from(context).notify(10, first);
        NotificationManagerCompat.from(context).notify(11, second);
        NotificationManagerCompat.from(context).notify(12, summary);
    }

    public static void cancelNotification(Context context, int id) {
        NotificationManagerCompat.from(context).cancel(id);
    }

    public static void cancelAllNotifications(Context context) {
        NotificationManagerCompat.from(context).cancelAll();
    }
}
```

到此为止，一个简单的 Android Wear 通知程序创建完毕。执行后会实现通知功能，如图 8-23 所示。

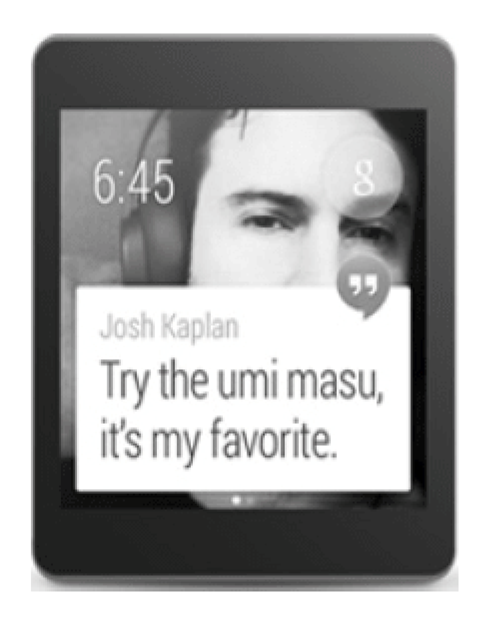


图 8-23　执行效果

有关 Android Wear 更多的演示程序，读者可以参考官方文档中的演示实例。

第 3 篇　实战演练篇

第 9 章　暴走轨迹计步器

第 10 章　智能家居系统

第 11 章　健康专家——智能心率计

第 12 章　湿度测试仪

第 13 章　小米录音机

第 14 章　智能楼宇灯光控制系统

第 15 章　智能闹钟系统

第 16 章　开发一个音乐播放器

第 17 章　移动阅读器系统

第 18 章　QR 码采集器

第 19 章　骑行记录仪



第 9 章　暴走轨迹计步器

暴走是指沿着指定路线徒步或驾车行走，时间不限。暴走这项运动源于美国，风靡欧美，是一种高强度又简单易行的户外运动方式，目前世界上暴走一族大约有 7000 万人。日渐流行的“暴走”，几乎成了时尚、健身、释放、减压的代名词。近年来随着全民健身热潮的兴起，为传感器应用开发提供了良好的舞台。本章将通过一个综合实例的实现过程，详细讲解利用 Android 传感器技术开发暴走轨迹计步器系统的方法。

9.1　系统功能模块介绍

知识点讲解：光盘:视频\知识点\第 9 章\系统功能模块介绍.avi

本章轨迹记录器的功能是，通过 Android 传感器记录当前的位置、速率、海拔、记录频率和距离等信息，并且可以将轨迹信息打包上传或分享。本章行走轨迹记录器的构成模块结构如图 9-1 所示。

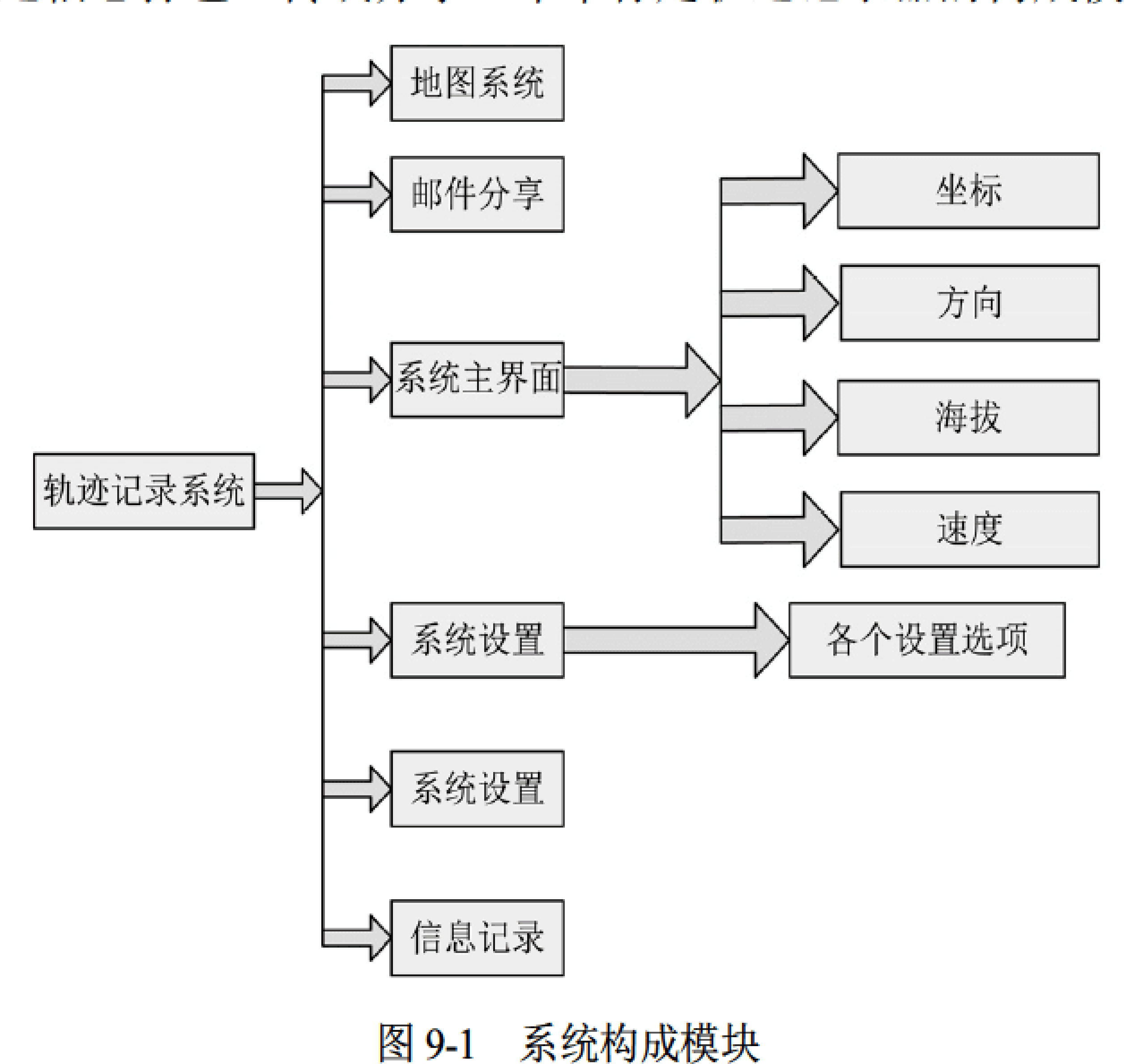


图 9-1　系统构成模块

9.2　系统主界面

知识点讲解：光盘:视频\知识点\第 9 章\系统主界面.avi

系统主界面是运行程序后首先呈现在用户面前的界面。本节将详细讲解本章行走轨迹记录器主界面的具体实现流程。

9.2.1　布局文件

本系统主界面的布局文件是 main.xml，功能是通过文本控件显示当前的位置信息和传感器信息，具体实现代码如下所示。

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/scroll" android:layout_width="fill_parent"
            android:layout_height="fill_parent" android:background="#000000">
    <LinearLayout
                android:layout_width="fill_parent" android:layout_height="fill_parent"
                android:orientation="vertical">
        <TextView android:id="@+id/textStatus" android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>
        <TableLayout android:id="@+id/TableGPS"
                    android:layout_width="fill_parent" android:layout_height="wrap_content"
                    android:stretchColumns="1" android:background="#000000">
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:id="@+id/txtDateTimeAndProvider"
                        android:gravity="left" android:textStyle="bold" android:padding="2dip"
                        android:layout_span="2"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:textStyle="bold" android:text="@string/txt_latitude"
                        android:padding="3dip" android:textSize="17sp"/>
                <TextView android:id="@+id/txtLatitude" android:gravity="left"
                        android:padding="3dip" android:textColor="#e8a317"
                        android:textStyle="bold" android:textSize="18sp"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:textStyle="bold" android:text="@string/txt_longitude"
                        android:padding="3dip" android:textSize="17sp"/>
                <TextView android:id="@+id/txtLongitude" android:gravity="left"
                        android:padding="3dip" android:textColor="#e8a317"
                        android:textStyle="bold" android:textSize="18sp"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:id="@+id/lblAltitude" android:textStyle="bold"
                        android:text="@string/txt_altitude" android:padding="3dip"/>
                <TextView android:id="@+id/txtAltitude" android:gravity="left"
                        android:padding="3dip"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:id="@+id/lblSpeed" android:textStyle="bold"
                        android:text="@string/txt_speed" android:padding="3dip"/>
                <TextView android:id="@+id/txtSpeed" android:gravity="left"
                        android:padding="3dip"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:id="@+id/lblDirection" android:textStyle="bold"
```

```
                    android:text="@string/txt_direction" android:padding="3dip"/>
            <TextView android:id="@+id/txtDirection" android:gravity="left"
                    android:padding="3dip"/>
        </TableRow>
        <TableRow android:background="#333333" android:layout_margin="1dip">
            <TextView android:id="@+id/lblSatellites" android:textStyle="bold"
                    android:text="@string/txt_satellites" android:padding="3dip"/>
            <TextView android:id="@+id/txtSatellites" android:gravity="left"
                    android:padding="3dip"/>
        </TableRow>
        <TableRow android:background="#333333" android:layout_margin="1dip">
            <TextView android:id="@+id/lblAccuracy" android:textStyle="bold"
                    android:text="@string/txt_accuracy" android:padding="3dip"/>
            <TextView android:id="@+id/txtAccuracy" android:gravity="left"
                    android:padding="3dip"/>
        </TableRow>
    </TableLayout>
    <LinearLayout android:orientation="vertical"
                android:layout_width="fill_parent" android:layout_height="wrap_content">
        <!--
            <Button android:id="@+id/buttonStart" android:layout_width="120px"
            android:layout_height="wrap_content" android:tag="Start"
            android:text="Start" /> <Button android:id="@+id/buttonStop"
            android:layout_width="120px" android:layout_height="wrap_content"
            android:tag="Stop" android:text="Stop" />
        -->
        <ToggleButton android:id="@+id/buttonOnOff "
                    android:layout_width="fill_parent" android:layout_height="wrap_content"
                    android:textOn="@string/btn_stop_logging"
android:textOff="@string/btn_start_logging"/>
    </LinearLayout>
    <TableLayout android:id="@+id/TableSummary"
                android:layout_width="fill_parent" android:layout_height="wrap_content"
                android:background="#222222">
        <TableRow android:layout_width="fill_parent" android:layout_height="fill_parent">
            <TextView android:id="@+id/lblLoggingTo" android:layout_width="wrap_content"
                    android:textSize="9dip" android:layout_height="fill_parent" android:textStyle="italic"
                    android:paddingLeft="8dip" android:text="@string/summary_loggingto"/>
            <TextView android:id="@+id/txtLoggingTo" android:layout_width="wrap_content"
                    android:paddingLeft="3dip" android:textSize="9dip" android:textStyle="italic"
                    android:layout_height="fill_parent"/>
        </TableRow>
        <TableRow android:layout_width="fill_parent"
                android:layout_height="fill_parent">
            <TextView android:id="@+id/lblFrequency" android:layout_width="wrap_content"
                    android:textSize="9dip" android:layout_height="fill_parent" android:textStyle="italic"
                    android:paddingLeft="8dip" android:text="@string/summary_freq_every"/>
            <TextView android:id="@+id/txtFrequency" android:layout_width="wrap_content"
                    android:paddingLeft="3dip" android:textSize="9dip" android:textStyle="italic"
                    android:layout_height="fill_parent"/>
        </TableRow>
```

```
            <TableRow android:layout_width="fill_parent"
                      android:layout_height="fill_parent">
                <TextView android:id="@+id/lblDistance" android:layout_width="wrap_content"
                          android:textSize="9dip" android:layout_height="fill_parent" android:textStyle="italic"
                          android:paddingLeft="8dip" android:text="@string/summary_dist"/>
                <TextView android:id="@+id/txtDistance" android:layout_width="wrap_content"
                          android:paddingLeft="3dip" android:textSize="9dip" android:textStyle="italic"
                          android:layout_height="fill_parent"/>
            </TableRow>
            <TableRow android:layout_width="fill_parent"
                      android:layout_height="fill_parent">
                <TextView android:id="@+id/lblFileName" android:layout_width="wrap_content"
                          android:textSize="9dip" android:layout_height="fill_parent" android:textStyle="italic"
                          android:paddingLeft="8dip" android:text="@string/summary_current_filename"/>
                <TextView android:id="@+id/txtFileName" android:layout_width="wrap_content"
                          android:paddingLeft="3dip" android:textSize="9dip" android:textStyle="italic"
                          android:layout_height="fill_parent"/>
            </TableRow>
            <TableRow android:layout_width="fill_parent"
                      android:layout_height="fill_parent" android:id="@+id/trAutoEmail">
                <TextView android:id="@+id/lblAutoEmail" android:layout_width="wrap_content"
                          android:textSize="9dip" android:layout_height="fill_parent" android:textStyle="italic"
                          android:paddingLeft="8dip" android:text="@string/summary_autoemail"/>
                <TextView android:id="@+id/txtAutoEmail" android:layout_width="wrap_content"
                          android:paddingLeft="3dip" android:textSize="9dip" android:textStyle="italic"
                         android:layout_height="fill_parent"/>
            </TableRow>
        </TableLayout>
        <!--    <TextView android:id="@+id/lblSummary" android:layout_width="fill_parent"
       android:layout_height="wrap_content" android:textStyle="italic"
       android:textSize="11dip" /> -->
    </LinearLayout>
</ScrollView>
```

9.2.2　实现主 Activity

本系统的主 Activity 是 GpsMainActivity，实现文件是 GpsMainActivity.java，具体实现流程如下。

（1）定义更新 UI 线程的类 GpsMainActivity，获取 ToggleButton 按钮的开关来显示位置信息，具体实现代码如下所示。

```
public class GpsMainActivity extends Activity implements OnCheckedChangeListener,
        IGpsLoggerServiceClient
{
    /**
     * 用处理器更新 UI 线程
     */
    public final Handler handler = new Handler();
    private static Intent serviceIntent;
    private GpsLoggingService loggingService;
    /**
```

```
         * 提供一个连接到 GPS 记录的服务
         */
        private ServiceConnection gpsServiceConnection = new ServiceConnection()
        {
                public void onServiceDisconnected(ComponentName name)
                {
                        loggingService = null;
                }
                public void onServiceConnected(ComponentName name, IBinder service)
                {
                        loggingService = ((GpsLoggingService.GpsLoggingBinder) service).getService();
                        GpsLoggingService.SetServiceClient(GpsMainActivity.this);
                        //设置切换按钮，显示现有的位置信息
                        ToggleButton buttonOnOff = (ToggleButton) findViewById(R.id.buttonOnOff);
                        if (Session.isStarted())
                        {
                                buttonOnOff.setChecked(true);
                                DisplayLocationInfo(Session.getCurrentLocationInfo());
                        }
                        buttonOnOff.setOnCheckedChangeListener(GpsMainActivity.this);
                }
        };
```

（2）定义第一次创建样式时触发的方法，具体实现代码如下所示。

```
/**
 * 第一次创建样式时触发的事件
 */
@Override
public void onCreate(Bundle savedInstanceState)
{
        SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        String lang = prefs.getString("locale_override", "");
        if (!lang.equalsIgnoreCase(""))
        {
                Locale locale = new Locale(lang);
                Locale.setDefault(locale);
                Configuration config = new Configuration();
                config.locale = locale;
                getBaseContext().getResources().updateConfiguration(config,
                                getBaseContext().getResources().getDisplayMetrics());
        }
        super.onCreate(savedInstanceState);
        Utilities.LogInfo("GPSLogger started");
        setContentView(R.layout.main);
        GetPreferences();
        StartAndBindService();
}
```

（3）启动定位服务并绑定到当前的 Activit 界面，具体实现代码如下所示。

```
private void StartAndBindService()
{
        Utilities.LogDebug("StartAndBindService - binding now");
```

```
        serviceIntent = new Intent(this, GpsLoggingService.class);
        // Start the service in case it isn't already running
        startService(serviceIntent);
        // Now bind to service
        bindService(serviceIntent, gpsServiceConnection, Context.BIND_AUTO_CREATE);
        Session.setBoundToService(true);
}
```

（4）当按钮关闭则停止系统的监听服务，具体实现代码如下所示。

```
private void StopAndUnbindServiceIfRequired()
{
        if(Session.isBoundToService())
        {
                unbindService(gpsServiceConnection);
                Session.setBoundToService(false);
        }
        if(!Session.isStarted())
        {
                Utilities.LogDebug("StopServiceIfRequired - Stopping the service");
                //serviceIntent = new Intent(this, GpsLoggingService.class);
                stopService(serviceIntent);
        }
}

@Override
protected void onPause()
{
        StopAndUnbindServiceIfRequired();
        super.onPause();
}

@Override
protected void onDestroy()
{
        StopAndUnbindServiceIfRequired();
        super.onDestroy();

}
```

（5）当切换按钮被单击时调用方法 onCheckedChanged，具体实现代码如下所示。

```
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
        if (isChecked)
        {
               GetPreferences();

                loggingService.StartLogging();
        }
        else
        {
                loggingService.StopLogging();
        }
}
```

（6）根据用户设置选项值显示一个具有良好可读性的视图界面，具体实现代码如下所示。

```
private void ShowPreferencesSummary()
{
	TextView txtLoggingTo = (TextView) findViewById(R.id.txtLoggingTo);
	TextView txtFrequency = (TextView) findViewById(R.id.txtFrequency);
	TextView txtDistance = (TextView) findViewById(R.id.txtDistance);
	TextView txtAutoEmail = (TextView) findViewById(R.id.txtAutoEmail);
	if (!AppSettings.shouldLogToKml() && !AppSettings.shouldLogToGpx())
	{
		txtLoggingTo.setText(R.string.summary_loggingto_screen);
	}
	else if (AppSettings.shouldLogToGpx() && AppSettings.shouldLogToKml())
	{
		txtLoggingTo.setText(R.string.summary_loggingto_both);
	}
	else
	{
		txtLoggingTo.setText((AppSettings.shouldLogToGpx() ? "GPX" : "KML"));
	}
	if (AppSettings.getMinimumSeconds() > 0)
	{
		String descriptiveTime = Utilities.GetDescriptiveTimeString(AppSettings.getMinimumSeconds(),
				getBaseContext());
		txtFrequency.setText(descriptiveTime);
	}
	else
	{
		txtFrequency.setText(R.string.summary_freq_max);
	}
	if (AppSettings.getMinimumDistance() > 0)
	{
		if (AppSettings.shouldUseImperial())
		{
			int minimumDistanceInFeet = Utilities.MetersToFeet(AppSettings.getMinimumDistance());
			txtDistance.setText(((minimumDistanceInFeet == 1)
				? getString(R.string.foot)
				: String.valueOf(minimumDistanceInFeet) + getString(R.string.feet)));
		}
		else
		{
			txtDistance.setText(((AppSettings.getMinimumDistance() == 1)
				? getString(R.string.meter)
				: String.valueOf(AppSettings.getMinimumDistance()) + getString(R.string.meters)));
		}
	}
	else
	{
		txtDistance.setText(R.string.summary_dist_regardless);
	}
```

```
        if (AppSettings.isAutoEmailEnabled())
        {
                String autoEmailResx;
                if (AppSettings.getAutoEmailDelay() == 0)
                {
                        autoEmailResx = "autoemail_frequency_whenistop";
                }
                else
                {

                        autoEmailResx = "autoemail_frequency_"
                                + String.valueOf(AppSettings.getAutoEmailDelay()).replace(".", "");
                        //.replace(".0", "")
                }
                String autoEmailDesc = getString(getResources().getIdentifier(autoEmailResx, "string", getPackage
Name()));

//                      String autoEmailDesc = getString(getResources().getIdentifier(
//                                      getPackageName() +   ":string/" + autoEmailResx, null, null));

                txtAutoEmail.setText(autoEmailDesc);
        }
        else
        {
                TableRow trAutoEmail = (TableRow) findViewById(R.id.trAutoEmail);
                trAutoEmail.setVisibility(View.INVISIBLE);
        }
}
```

（7）根据用户选择的菜单项调用执行不同的处理方法，具体实现代码如下所示。

```
public boolean onOptionsItemSelected(MenuItem item)
{
        int itemId = item.getItemId();
        Utilities.LogInfo("Option item selected - " + String.valueOf(item.getTitle()));
        switch (itemId)
        {
                case R.id.mnuSettings:
                        Intent settingsActivity = new Intent(getBaseContext(), GpsSettingsActivity.class);
                        startActivity(settingsActivity);
                        break;
                case R.id.mnuOSM:
                        UploadToOpenStreetMap();
                        break;
                case R.id.mnuAnnotate:
                        Annotate();
                        break;
                case R.id.mnuShare:
                        Share();
                        break;
                case R.id.mnuEmailnow:
                        EmailNow();
```

```
                break;
            case R.id.mnuExit:
                loggingService.StopLogging();
                loggingService.stopSelf();
                System.exit(0);
                break;
        }
        return false;
}
private void EmailNow()
{
        if(Utilities.IsEmailSetup(getBaseContext()))
        {
            loggingService.ForceEmailLogFile();
        }
        else
        {
            Intent emailSetup = new Intent(getBaseContext(), AutoEmailActivity.class);
            startActivity(emailSetup);
        }
}
```

（8）通过方法 Share 实现轨迹分享功能，允许用户发送带位置的 GPX/KML 文件的位置，或者只使用一个提供者，可以使用的分享方式有 Facebook、短信、电子邮件、推特和蓝牙。方法 Share 的具体实现代码如下所示。

```
private void Share()
{
        try
        {
            final String locationOnly = getString(R.string.sharing_location_only);
            final File gpxFolder = new File(Environment.getExternalStorageDirectory(), "GPSLogger");
            if (gpxFolder.exists())
            {
                String[ ] enumeratedFiles = gpxFolder.list();
                List<String> fileList = new ArrayList<String>(Arrays.asList(enumeratedFiles));
                Collections.reverse(fileList);
                fileList.add(0, locationOnly);
                final String[ ] files = fileList.toArray(new String[0]);
                final Dialog dialog = new Dialog(this);
                dialog.setTitle(R.string.sharing_pick_file);
                dialog.setContentView(R.layout.filelist);
                ListView thelist = (ListView) dialog.findViewById(R.id.listViewFiles);
                thelist.setAdapter(new ArrayAdapter<String>(getBaseContext(),
                        android.R.layout.simple_list_item_single_choice, files));
                thelist.setOnItemClickListener(new OnItemClickListener()
                {
                    public void onItemClick(AdapterView<?> av, View v, int index, long arg)
                    {
                        dialog.dismiss();
                        String chosenFileName = files[index];
                        final Intent intent = new Intent(Intent.ACTION_SEND);
```

```
                    // intent.setType("text/plain");
                    intent.setType("*/*");
                    if (chosenFileName.equalsIgnoreCase(locationOnly))
                    {
                        intent.setType("text/plain");
                    }
                    intent.putExtra(Intent.EXTRA_SUBJECT, getString(R.string.sharing_mylocation));
                    if (Session.hasValidLocation())
                    {
                        String bodyText = getString(R.string.sharing_latlong_text,
                                String.valueOf(Session.getCurrentLatitude()),
                                String.valueOf(Session.getCurrentLongitude()));
                        intent.putExtra(Intent.EXTRA_TEXT, bodyText);
                        intent.putExtra("sms_body", bodyText);
                    }
                    if (chosenFileName.length() > 0
                            && !chosenFileName.equalsIgnoreCase(locationOnly))
                    {
                        intent.putExtra(Intent.EXTRA_STREAM,
                                Uri.fromFile(new File(gpxFolder, chosenFileName)));
                    }
                    startActivity(Intent.createChooser(intent, getString(R.string.sharing_via)));
                }
            });
            dialog.show();
        }
        else
        {
            Utilities.MsgBox(getString(R.string.sorry), getString(R.string.no_files_found), this);
        }
    }
    catch (Exception ex)
    {
        Utilities.LogError("Share", ex);
    }
}
```

（9）编写方法 UploadToOpenStreetMap 上传一个跟踪 GPS 记录的对象，具体实现代码如下所示。

```
private void UploadToOpenStreetMap()
{
    if(!Utilities.IsOsmAuthorized(getBaseContext()))
    {
        startActivity(Utilities.GetOsmSettingsIntent(getBaseContext()));
        return;
    }
    final String goToOsmSettings = getString(R.string.menu_settings);

    final File gpxFolder = new File(Environment.getExternalStorageDirectory(), "GPSLogger");
    if (gpxFolder.exists())
    {
        FilenameFilter select = new FilenameFilter()
```

```
            {
                  public boolean accept(File dir, String filename)
                  {
                      return filename.toLowerCase().contains(".gpx");
                  }
            };
            String[ ] enumeratedFiles = gpxFolder.list(select);
            List<String> fileList = new ArrayList<String>(Arrays.asList(enumeratedFiles));
            Collections.reverse(fileList);
            fileList.add(0, goToOsmSettings);
            final String[ ] files = fileList.toArray(new String[0]);
            final Dialog dialog = new Dialog(this);
            dialog.setTitle(R.string.osm_pick_file);
            dialog.setContentView(R.layout.filelist);
            ListView thelist = (ListView) dialog.findViewById(R.id.listViewFiles);
            thelist.setAdapter(new ArrayAdapter<String>(getBaseContext(),
                        android.R.layout.simple_list_item_single_choice, files));
            thelist.setOnItemClickListener(new OnItemClickListener()
            {
                  public void onItemClick(AdapterView<?> av, View v, int index, long arg)
                  {

                        dialog.dismiss();
                        String chosenFileName = files[index];

                        if(chosenFileName.equalsIgnoreCase(goToOsmSettings))
                        {
                              startActivity(Utilities.GetOsmSettingsIntent(getBaseContext()));
                        }
                        else
                        {
                              OSMHelper osm = new OSMHelper(GpsMainActivity.this);
                              Utilities.ShowProgress(GpsMainActivity.this, getString(R.string.osm_uploading), getString
(R.string.please_wait));
                              osm.UploadGpsTrace(chosenFileName);
                        }
                  }
            });
            dialog.show();
      }
      else
      {
            Utilities.MsgBox(getString(R.string.sorry), getString(R.string.no_files_found), this);
      }
}
```

（10）通过方法 Annotate 提示用户输入，然后添加文本日志文件，具体实现代码如下所示。

```
private void Annotate()
{
      if (!AppSettings.shouldLogToGpx() && !AppSettings.shouldLogToKml())
```

```
    {
        return;
    }

    if (!Session.shoulAllowDescription())
    {
        Utilities.MsgBox(getString(R.string.not_yet),
                getString(R.string.cant_add_description_until_next_point),
                GetActivity());
        return;
    }
    AlertDialog.Builder alert = new AlertDialog.Builder(GpsMainActivity.this);
    alert.setTitle(R.string.add_description);
    alert.setMessage(R.string.letters_numbers);
    //设置一个 EditText 视图用来获取用户的输入
    final EditText input = new EditText(getBaseContext());
    alert.setView(input);
    alert.setPositiveButton(R.string.ok, new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int whichButton)
        {
            final String desc = Utilities.CleanDescription(input.getText().toString());
            Annotate(desc);
        }
    });
    alert.setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int whichButton)
        {
            // Cancelled.
        }
    });
    alert.show();
}
```

（11）编写方法 ClearForm 清理当前屏幕视图，并删除所有获取的值。具体实现代码如下所示。

```
public void ClearForm()
{
    TextView tvLatitude = (TextView) findViewById(R.id.txtLatitude);
    TextView tvLongitude = (TextView) findViewById(R.id.txtLongitude);
    TextView tvDateTime = (TextView) findViewById(R.id.txtDateTimeAndProvider);
    TextView tvAltitude = (TextView) findViewById(R.id.txtAltitude);
    TextView txtSpeed = (TextView) findViewById(R.id.txtSpeed);
    TextView txtSatellites = (TextView) findViewById(R.id.txtSatellites);
    TextView txtDirection = (TextView) findViewById(R.id.txtDirection);
    TextView txtAccuracy = (TextView) findViewById(R.id.txtAccuracy);
    tvLatitude.setText("");
    tvLongitude.setText("");
    tvDateTime.setText("");
    tvAltitude.setText("");
    txtSpeed.setText("");
```

```
        txtSatellites.setText("");
        txtDirection.setText("");
        txtAccuracy.setText("");
}
```

（12）在顶部的状态标签设置信息，具体实现代码如下所示。

```
private void SetStatus(String message)
{
        TextView tvStatus = (TextView) findViewById(R.id.textStatus);
        tvStatus.setText(message);
        Utilities.LogInfo(message);
}
```

（13）设置表中的卫星视图，具体实现代码如下所示。

```
private void SetSatelliteInfo(int number)
{
        Session.setSatelliteCount(number);
        TextView txtSatellites = (TextView) findViewById(R.id.txtSatellites);
        txtSatellites.setText(String.valueOf(number));
}
```

（14）处理指定的位置坐标，并将结果显示在视图中。具体实现代码如下所示。

```
private void DisplayLocationInfo(Location loc)
{
        try
        {
                if (loc == null)
                {
                        return;
                }
                Session.setLatestTimeStamp(System.currentTimeMillis());
                TextView tvLatitude = (TextView) findViewById(R.id.txtLatitude);
                TextView tvLongitude = (TextView) findViewById(R.id.txtLongitude);
                TextView tvDateTime = (TextView) findViewById(R.id.txtDateTimeAndProvider);
                TextView tvAltitude = (TextView) findViewById(R.id.txtAltitude);
                TextView txtSpeed = (TextView) findViewById(R.id.txtSpeed);
                TextView txtSatellites = (TextView) findViewById(R.id.txtSatellites);
                TextView txtDirection = (TextView) findViewById(R.id.txtDirection);
                TextView txtAccuracy = (TextView) findViewById(R.id.txtAccuracy);
                String providerName = loc.getProvider();
                if (providerName.equalsIgnoreCase("gps"))
                {
                        providerName = getString(R.string.providername_gps);
                }
                else
                {
                        providerName = getString(R.string.providername_celltower);
                }
                tvDateTime.setText(new Date().toLocaleString()
                                + getString(R.string.providername_using, providerName));
                tvLatitude.setText(String.valueOf(loc.getLatitude()));
                tvLongitude.setText(String.valueOf(loc.getLongitude()));
                if (loc.hasAltitude())
```

```
{
    double altitude = loc.getAltitude();
    if (AppSettings.shouldUseImperial())
    {
        tvAltitude.setText(String.valueOf(Utilities.MetersToFeet(altitude))
                + getString(R.string.feet));
    }
    else
    {
        tvAltitude.setText(String.valueOf(altitude) + getString(R.string.meters));
    }
}
else
{
    tvAltitude.setText(R.string.not_applicable);
}           if (loc.hasSpeed())
{
    float speed = loc.getSpeed();
    if (AppSettings.shouldUseImperial())
    {
        txtSpeed.setText(String.valueOf(Utilities.MetersToFeet(speed))
                + getString(R.string.feet_per_second));
    }
    else
    {
        txtSpeed.setText(String.valueOf(speed) + getString(R.string.meters_per_second));
    }
}
else
{
    txtSpeed.setText(R.string.not_applicable);
}
if (loc.hasBearing())
{
    float bearingDegrees = loc.getBearing();
    String direction;
    direction = Utilities.GetBearingDescription(bearingDegrees, getBaseContext());
    txtDirection.setText(direction + "(" + String.valueOf(Math.round(bearingDegrees))
            + getString(R.string.degree_symbol) + ")");
}
else
{
    txtDirection.setText(R.string.not_applicable);
}
if (!Session.isUsingGps())
{
    txtSatellites.setText(R.string.not_applicable);
    Session.setSatelliteCount(0);
}
if (loc.hasAccuracy())
{
```

```
                float accuracy = loc.getAccuracy();
                if (AppSettings.shouldUseImperial())
                {
                    txtAccuracy.setText(getString(R.string.accuracy_within,
                            String.valueOf(Utilities.MetersToFeet(accuracy)), getString(R.string.feet)));
                }
                else
                {
                    txtAccuracy.setText(getString(R.string.accuracy_within, String.valueOf(accuracy),
                            getString(R.string.meters)));
                }
            }
            else
            {
                txtAccuracy.setText(R.string.not_applicable);
            }
        }
        catch (Exception ex)
        {
            SetStatus(getString(R.string.error_displaying, ex.getMessage()));
        }
}
```

在主 Activity 的实现过程中用到了系统服务 Activity，其实现文件是 GpsLoggingService.java，功能是提供了本系统所需要的后台服务方法。文件 GpsLoggingService.java 的具体实现流程如下所示。

（1）定义可以调用的类和方法，具体实现代码如下所示。

```
class GpsLoggingBinder extends Binder
{
    public GpsLoggingService getService()
    {
        Utilities.LogDebug("GpsLoggingBinder.getService");
        return GpsLoggingService.this;
    }
}
```

（2）建立基于用户偏好设置的电邮自动计时器，具体实现代码如下所示。

```
private void SetupAutoEmailTimers()
{
    Utilities.LogDebug("GpsLoggingService.SetupAutoEmailTimers");
    Utilities.LogDebug("isAutoEmailEnabled - " + String.valueOf(AppSettings.isAutoEmailEnabled()));
    Utilities.LogDebug("Session.getAutoEmailDelay - " + String.valueOf(Session.getAutoEmailDelay()));
    if (AppSettings.isAutoEmailEnabled() && Session.getAutoEmailDelay() > 0)
    {
        Utilities.LogDebug("Setting up email alarm");
        long triggerTime = System.currentTimeMillis()
                + (long) (Session.getAutoEmailDelay() * 60 * 60 * 1000);
        alarmIntent = new Intent(getBaseContext(), AlarmReceiver.class);
        PendingIntent sender = PendingIntent.getBroadcast(this, 0, alarmIntent,
                PendingIntent.FLAG_UPDATE_CURRENT);
        AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
        am.set(AlarmManager.RTC_WAKEUP, triggerTime, sender);
    }
```

```
        else
        {
            Utilities.LogDebug("Checking if alarmIntent is null");
            if (alarmIntent != null)
            {
                Utilities.LogDebug("alarmIntent was null, canceling alarm");
                CancelAlarm();
            }
        }
}
```

（3）如果调用用户选择自动邮件日志文件方法则停止记录操作，具体实现代码如下所示。

```
private void AutoEmailLogFileOnStop()
{
        Utilities.LogDebug("GpsLoggingService.AutoEmailLogFileOnStop");
        Utilities.LogVerbose("isAutoEmailEnabled - " + AppSettings.isAutoEmailEnabled());
        // autoEmailDelay 0 means send it when you stop logging.
        if (AppSettings.isAutoEmailEnabled() && Session.getAutoEmailDelay() == 0)
        {
            Session.setEmailReadyToBeSent(true);
            AutoEmailLogFile();
        }
}
```

（4）调用自动电子邮件辅助处理文件并将其发送，具体实现代码如下所示。

```
private void AutoEmailLogFile()
{
        Utilities.LogDebug("GpsLoggingService.AutoEmailLogFile");
        Utilities.LogVerbose("isEmailReadyToBeSent - " + Session.isEmailReadyToBeSent());
        if (Session.getCurrentFileName() != null && Session.getCurrentFileName().length() > 0
                && Session.isEmailReadyToBeSent())
        {
            if(IsMainFormVisible())
            {
                Utilities.ShowProgress(mainServiceClient.GetActivity(), getString(R.string.autoemail_sending),
                    getString(R.string.please_wait));
            }
            Utilities.LogInfo("Emailing Log File");
            AutoEmailHelper aeh = new AutoEmailHelper(GpsLoggingService.this);
            aeh.SendLogFile(Session.getCurrentFileName(), false);
            SetupAutoEmailTimers();

            if(IsMainFormVisible())
            {
                Utilities.HideProgress();
            }
        }
}
protected void ForceEmailLogFile()
{

        Utilities.LogDebug("GpsLoggingService.ForceEmailLogFile");
```

```
        if (Session.getCurrentFileName() != null && Session.getCurrentFileName().length() > 0)
        {
            if(IsMainFormVisible())
            {
                Utilities.ShowProgress(mainServiceClient.GetActivity(), getString(R.string.autoemail_sending),
                    getString(R.string.please_wait));
            }
            Utilities.LogInfo("Force emailing Log File");
            AutoEmailHelper aeh = new AutoEmailHelper(GpsLoggingService.this);
            aeh.SendLogFile(Session.getCurrentFileName(), true);

            if(IsMainFormVisible())
            {
                Utilities.HideProgress();
            }
        }
}
```

（5）设置此服务的活动形式，活动形式需要调用 igpsloggerserviceclient，具体实现代码如下所示。

```
protected static void SetServiceClient(IGpsLoggerServiceClient mainForm)
{
    mainServiceClient = mainForm;
}
```

（6）根据用户的偏好设置选择和填充 appSettings 对象，并设置电子邮件的定时器，具体实现代码如下所示。

```
private void GetPreferences()
{
    Utilities.LogDebug("GpsLoggingService.GetPreferences");
    Utilities.PopulateAppSettings(getBaseContext());
    Utilities.LogDebug("Session.getAutoEmailDelay: " + Session.getAutoEmailDelay());
    Utilities.LogDebug("AppSettings.getAutoEmailDelay: " + AppSettings.getAutoEmailDelay());
    if (Session.getAutoEmailDelay() != AppSettings.getAutoEmailDelay())
    {
        Utilities.LogDebug("Old autoEmailDelay - " + String.valueOf(Session.getAutoEmailDelay())
                + "; New -" + String.valueOf(AppSettings.getAutoEmailDelay()));
        Session.setAutoEmailDelay(AppSettings.getAutoEmailDelay());
        SetupAutoEmailTimers();
    }
}
```

（7）实现复位处理，具体实现代码如下所示。

```
protected void StartLogging()
{
    Utilities.LogDebug("GpsLoggingService.StartLogging");
    Session.setAddNewTrackSegment(true);
    if (Session.isStarted())
    {
        return;
    }
    Utilities.LogInfo("Starting logging procedures");
    startForeground(NOTIFICATION_ID, null);
    Session.setStarted(true);
```

```
        GetPreferences();
        Notify();
        ResetCurrentFileName();
        ClearForm();
        StartGpsManager();
}
```

（8）停止记录，删除通知，停止 GPS 经理，通过定时器停止邮件，具体实现代码如下所示。

```
protected void StopLogging()
{
        Utilities.LogDebug("GpsLoggingService.StopLogging");
        Session.setAddNewTrackSegment(true);
        Utilities.LogInfo("Stopping logging");
        Session.setStarted(false);
        // Email log file before setting location info to null
        AutoEmailLogFileOnStop();
        CancelAlarm();
        Session.setCurrentLocationInfo(null);
        stopForeground(true);
        RemoveNotification();
        StopGpsManager();
        StopMainActivity();
}
```

（9）在状态栏中显示通知，具体实现代码如下所示。

```
private void Notify()
{
        Utilities.LogDebug("GpsLoggingService.Notify");
        if (AppSettings.shouldShowInNotificationBar())
        {
                gpsNotifyManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
                ShowNotification();
        }
        else
        {
                RemoveNotification();
        }
}
```

（10）如果图标是可见的则隐藏状态栏中的通知，具体实现代码如下所示。

```
private void RemoveNotification()
{
        Utilities.LogDebug("GpsLoggingService.RemoveNotification");
        try
        {
                if (Session.isNotificationVisible())
                {
                        gpsNotifyManager.cancelAll();
                }
        }
        catch (Exception ex)
        {
                Utilities.LogError("RemoveNotification", ex);
```

```
        }
        finally
        {
            // notificationVisible = false;
            Session.setNotificationVisible(false);
        }
}
```

（11）在状态栏中显示 GPS 记录器的通知图标，具体实现代码如下所示。

```
private void ShowNotification()
{
        Utilities.LogDebug("GpsLoggingService.ShowNotification");
        // What happens when the notification item is clicked
        Intent contentIntent = new Intent(this, GpsMainActivity.class);
        PendingIntent pending = PendingIntent.getActivity(getBaseContext(), 0, contentIntent,
                    android.content.Intent.FLAG_ACTIVITY_NEW_TASK);
        Notification nfc = new Notification(R.drawable.gpsloggericon2, null, System.currentTimeMillis());
        nfc.flags |= Notification.FLAG_ONGOING_EVENT;
        NumberFormat nf = new DecimalFormat("###.######");
        String contentText = getString(R.string.gpslogger_still_running);
        if (Session.hasValidLocation())
        // if (currentLatitude != 0 && currentLongitude != 0)
        {
            contentText = nf.format(Session.getCurrentLatitude()) + ","
                        + nf.format(Session.getCurrentLongitude());
        }
        nfc.setLatestEventInfo(getBaseContext(), getString(R.string.gpslogger_still_running),
                    contentText, pending);
        gpsNotifyManager.notify(NOTIFICATION_ID, nfc);
        Session.setNotificationVisible(true);
}
```

（12）根据用户的偏好设置选项启动 GPS 功能，具体实现代码如下所示。

```
private void StartGpsManager()
{
        Utilities.LogDebug("GpsLoggingService.StartGpsManager");
        GetPreferences();
        gpsLocationListener = new GeneralLocationListener(this);
        towerLocationListener = new GeneralLocationListener(this);
        gpsLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        towerLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        CheckTowerAndGpsStatus();
        if (Session.isGpsEnabled() && !AppSettings.shouldPreferCellTower())
        {
            Utilities.LogInfo("Requesting GPS location updates");
            // gps satellite based
            gpsLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
                        AppSettings.getMinimumSeconds() * 1000, AppSettings.getMinimumDistance(),
                        gpsLocationListener);
            gpsLocationManager.addGpsStatusListener(gpsLocationListener);
            Session.setUsingGps(true);
        }
```

```
        else if (Session.isTowerEnabled())
        {
            Utilities.LogInfo("Requesting tower location updates");
            Session.setUsingGps(false);
            // isUsingGps = false;
            // Cell tower and wifi based
            towerLocationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
                    AppSettings.getMinimumSeconds() * 1000, AppSettings.getMinimumDistance(),
                    towerLocationListener);
        }
        else
        {
            Utilities.LogInfo("No provider available");
            Session.setUsingGps(false);
            SetStatus(R.string.gpsprovider_unavailable);
            SetFatalMessage(R.string.gpsprovider_unavailable);
            StopLogging();
            return;
        }
        SetStatus(R.string.started);
}
```

（13）周期性检查是否已经启动 GPS 和信号塔，具体实现代码如下所示。

```
private void CheckTowerAndGpsStatus()
{
Session.setTowerEnabled(towerLocationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER));
Session.setGpsEnabled(gpsLocationManager.isProviderEnabled(LocationManager.GPS_PROVIDER));
}
```

（14）停止位置管理服务，具体实现代码如下所示。

```
private void StopGpsManager()
{
        Utilities.LogDebug("GpsLoggingService.StopGpsManager");
        if (towerLocationListener != null)
        {
            towerLocationManager.removeUpdates(towerLocationListener);
        }
        if (gpsLocationListener != null)
        {
            gpsLocationManager.removeUpdates(gpsLocationListener);
            gpsLocationManager.removeGpsStatusListener(gpsLocationListener);
        }
        SetStatus(getString(R.string.stopped));
}
```

（15）基于用户偏好设置当前文件名，具体实现代码如下所示。

```
private void ResetCurrentFileName()
{
        Utilities.LogDebug("GpsLoggingService.ResetCurrentFileName");
        String newFileName;
        if (AppSettings.shouldCreateNewFileOnceADay())
        {
            // 20100114.gpx
```

```
            SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");
            newFileName = sdf.format(new Date());
            Session.setCurrentFileName(newFileName);
        }
        else
        {
            // 20100114183329.gpx
            SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
            newFileName = sdf.format(new Date());
            Session.setCurrentFileName(newFileName);
        }
        if (IsMainFormVisible())
        {
            mainServiceClient.onFileName(newFileName);
        }
}
```

（16）为客户端显示一个状态信息，具体实现代码如下所示。

```
void SetStatus(String status)
{
        if (IsMainFormVisible())
        {
            mainServiceClient.OnStatusMessage(status);
        }
}
```

到此为止，系统的主 Activity 和服务 Activity 的实现过程介绍完毕，执行后的效果如图 9-2 所示。点击设备中的 MENU 按钮后，在屏幕下方弹出选项设置界面，如图 9-3 所示。



图 9-2　系统主界面

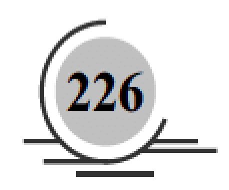


图 9-3　屏幕下方弹出选项设置界面

9.3　系统设置

知识点讲解：光盘:视频\知识点\第 9 章\系统设置.avi

当选择图 9-3 中的 Settings 选项后，会弹出系统设置界面，如图 9-4 所示。



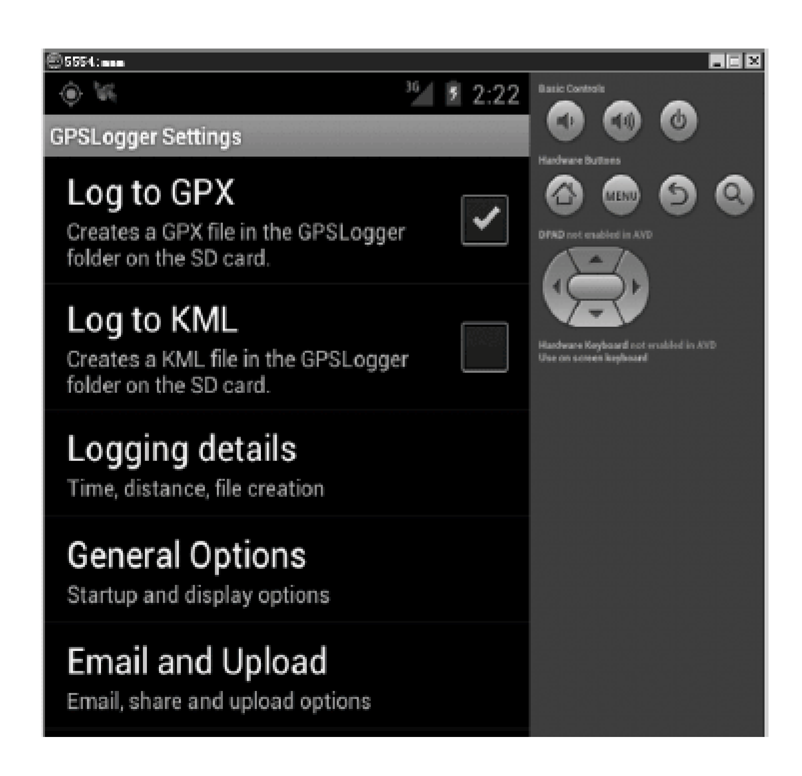


图 9-4　系统设置界面

在系统设置界面中，可以设置系统的常用选项参数。在本节的内容中，将详细讲解系统设置模块的实现过程。

9.3.1　选项设置

编写文件 AppSettings.java，功能是根据用户各个选项的值来设置系统，例如设置保存为 GPX（GPS eXchange Format 的缩写，译为 GPS 交换格式，是一个 XML 格式，为应用软件设计的通用 GPS 数据格式，可以用来描述路点、轨迹、路程）格式数据文件或 KML（是一种文件格式，用于在地球浏览器中显示地理数据，例如 Google 地球、Google 地图和谷歌手机地图）格式数据文件。文件 AppSettings.java 的具体实现代码如下所示。

```
public class AppSettings extends Application
{
    // ----------------------------------------------
    //用户设置
    // ----------------------------------------------
    private static boolean useImperial = false;
    private static boolean newFileOnceADay;
    private static boolean preferCellTower;
    private static boolean useSatelliteTime;
    private static boolean logToKml;
    private static boolean logToGpx;
    private static boolean showInNotificationBar;
    private static int minimumDistance;
    private static int minimumSeconds;
    private static String newFileCreation;
    private static Float autoEmailDelay = 0f;
    private static boolean autoEmailEnabled = false;
    private static String smtpServer;
    private static String smtpPort;
    private static String smtpUsername;
    private static String smtpPassword;
    private static String autoEmailTarget;
    private static boolean smtpSsl;
    private static boolean debugToFile;
    public static boolean shouldUseImperial()
    {
```

```
        return useImperial;
    }
    static void setUseImperial(boolean useImperial)
    {
        AppSettings.useImperial = useImperial;
    }
    /**
     * @return the 一天更新一个新文件
     */
    public static boolean shouldCreateNewFileOnceADay()
    {
        return newFileOnceADay;
    }
    static void setNewFileOnceADay(boolean newFileOnceADay)
    {
        AppSettings.newFileOnceADay = newFileOnceADay;
    }
    public static boolean shouldPreferCellTower()
    {
        return preferCellTower;
    }
    static void setPreferCellTower(boolean preferCellTower)
    {
        AppSettings.preferCellTower = preferCellTower;
    }
    public static boolean shouldUseSatelliteTime()
    {
        return useSatelliteTime;
    }
    static void setUseSatelliteTime(boolean useSatelliteTime)
    {
        AppSettings.useSatelliteTime = useSatelliteTime;
    }
    public static boolean shouldLogToKml()
    {
        return logToKml;
    }
    static void setLogToKml(boolean logToKml)
    {
        AppSettings.logToKml = logToKml;
    }
    public static boolean shouldLogToGpx()
    {
        return logToGpx;
    }
    static void setLogToGpx(boolean logToGpx)
    {
        AppSettings.logToGpx = logToGpx;
    }
    public static boolean shouldShowInNotificationBar()
    {
```

```
        return showInNotificationBar;
    }
    static void setShowInNotificationBar(boolean showInNotificationBar)
    {
        AppSettings.showInNotificationBar = showInNotificationBar;
    }
    public static int getMinimumDistance()
    {
        return minimumDistance;
    }
    static void setMinimumDistance(int minimumDistance)
    {
        AppSettings.minimumDistance = minimumDistance;
    }
    public static int getMinimumSeconds()
    {
        return minimumSeconds;
    }

    /**
     * @param minimumSeconds
     *            the minimumSeconds to set
     */
    static void setMinimumSeconds(int minimumSeconds)
    {
        AppSettings.minimumSeconds = minimumSeconds;
    }
```

9.3.2 生成 GPX 文件和 KML 文件

在系统设置界面中，可以指定一个文件来保存我们的行走轨迹。本系统提供了两种保存轨迹的文件格式，分别是 GPX 和 KML，如图 9-5 所示。

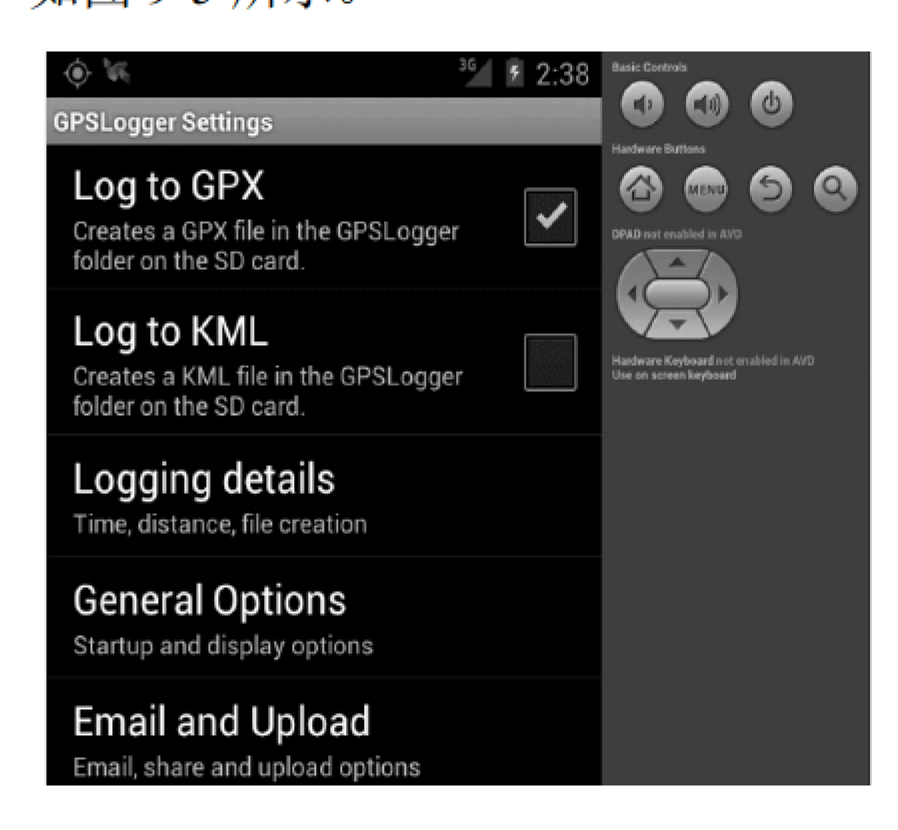


图 9-5 设置保存轨迹的文件格式

（1）编写文件 Gpx10FileLogger.java 生成 GPX 格式的文件，具体实现代码如下所示。

```
class Gpx10FileLogger implements IFileLogger
{
    private final static Object lock = new Object();
```

```
    private File gpxFile = null;
    private boolean useSatelliteTime = false;
    private boolean addNewTrackSegment;
   private int satelliteCount;

    Gpx10FileLogger(File gpxFile, boolean useSatelliteTime, boolean addNewTrackSegment, int satelliteCount)
    {
            this.gpxFile = gpxFile;
            this.useSatelliteTime = useSatelliteTime;
            this.addNewTrackSegment = addNewTrackSegment;
          this.satelliteCount = satelliteCount;
    }
    public void Write(Location loc) throws Exception
    {
            try
            {
                    Date now;
                    if (useSatelliteTime)
                    {
                            now = new Date(loc.getTime());
                    }
                    else
                    {
                            now = new Date();
                    }
                    String dateTimeString = Utilities.GetIsoDateTime(now);

                    if (!gpxFile.exists())
                    {
                            gpxFile.createNewFile();
                            FileOutputStream initialWriter = new FileOutputStream(gpxFile, true);
                            BufferedOutputStream initialOutput = new BufferedOutputStream(initialWriter);
                            String initialXml = "<?xml version=\"1.0\"?>"
                                            + "<gpx version=\"1.0\" creator=\"GPSLogger - http://gpslogger.mendhak.com/\" "
                                    + "xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" "
                                    + "xmlns=\"http://www.topografix.com/GPX/1/0\" "
                                    + "xsi:schemaLocation=\"http://www.topografix.com/GPX/1/0 "
                                    + "http://www.topografix.com/GPX/1/0/gpx.xsd\">"
                                            + "<time>" + dateTimeString + "</time>" + "<bounds />" + "<trk></trk></gpx>";
                            initialOutput.write(initialXml.getBytes());
                            initialOutput.flush();
                            initialOutput.close();
                    }
                DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
                DocumentBuilder builder = factory.newDocumentBuilder();
                Document doc = builder.parse(gpxFile);
                Node trkSegNode;
                NodeList trkSegNodeList = doc.getElementsByTagName("trkseg");
                if(addNewTrackSegment || trkSegNodeList.getLength()==0)
                {
                    NodeList trkNodeList = doc.getElementsByTagName("trk");
```

```
        trkSegNode = doc.createElement("trkseg");
        trkNodeList.item(0).appendChild(trkSegNode);
    }
    else
    {
        trkSegNode = trkSegNodeList.item(trkSegNodeList.getLength()-1);
    }
    Element trkptNode = doc.createElement("trkpt");
    Attr latAttribute = doc.createAttribute("lat");
    latAttribute.setValue(String.valueOf(loc.getLatitude()));
    trkptNode.setAttributeNode(latAttribute);
    Attr lonAttribute = doc.createAttribute("lon");
    lonAttribute.setValue(String.valueOf(loc.getLongitude()));
    trkptNode.setAttributeNode(lonAttribute);
    if(loc.hasAltitude())
    {
        Node eleNode = doc.createElement("ele");
        eleNode.appendChild(doc.createTextNode(String.valueOf(loc.getAltitude())));
        trkptNode.appendChild(eleNode);
    }
    Node timeNode = doc.createElement("time");
    timeNode.appendChild(doc.createTextNode(dateTimeString));
    trkptNode.appendChild(timeNode);
    trkSegNode.appendChild(trkptNode);
    if(loc.hasBearing())
    {
        Node courseNode = doc.createElement("course");
        courseNode.appendChild(doc.createTextNode(String.valueOf(loc.getBearing())));
        trkptNode.appendChild(courseNode);
    }
    if(loc.hasSpeed())
    {
        Node speedNode = doc.createElement("speed");
        speedNode.appendChild(doc.createTextNode(String.valueOf(loc.getSpeed())));
        trkptNode.appendChild(speedNode);
    }
    Node srcNode = doc.createElement("src");
    srcNode.appendChild(doc.createTextNode(loc.getProvider()));
    trkptNode.appendChild(srcNode);
    if(Session.getSatelliteCount() > 0)
    {
        Node satNode = doc.createElement("sat");
        satNode.appendChild(doc.createTextNode(String.valueOf(satelliteCount)));
        trkptNode.appendChild(satNode);
    }
    String newFileContents = Utilities.GetStringFromNode(doc);
    synchronized(lock)
    {
        FileOutputStream fos = new FileOutputStream(gpxFile, false);
        fos.write(newFileContents.getBytes());
        fos.close();
```

```
                }
            }
            catch (Exception e)
            {
                Utilities.LogError("Gpx10FileLogger.Write", e);
                throw new Exception("Could not write to GPX file");
            }
        }
        public void Annotate(String description) throws Exception
        {
            if (!gpxFile.exists())
            {
                return;
            }
            try
            {
                DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
                DocumentBuilder builder = factory.newDocumentBuilder();
                Document doc = builder.parse(gpxFile);
                NodeList trkptNodeList = doc.getElementsByTagName("trkpt");
                Node lastTrkPt = trkptNodeList.item(trkptNodeList.getLength()-1);
                Node nameNode = doc.createElement("name");
                nameNode.appendChild(doc.createTextNode(description));
                lastTrkPt.appendChild(nameNode);
                Node descNode = doc.createElement("desc");
                descNode.appendChild(doc.createTextNode(description));
                lastTrkPt.appendChild(descNode);
                String newFileContents = Utilities.GetStringFromNode(doc);
                synchronized(lock)
                {
                    FileOutputStream fos = new FileOutputStream(gpxFile, false);
                    fos.write(newFileContents.getBytes());
                    fos.close();
                }
            }
            catch(Exception e)
            {
                Utilities.LogError("Gpx10FileLogger.Annotate", e);
                throw new Exception("Could not annotate GPX file");
            }
        }
}
```

（2）编写文件 Kml10FileLogger.java 生成 KML 格式文件，具体实现代码如下所示。

```
class Kml10FileLogger implements IFileLogger
{
    private final static Object lock = new Object();
    private boolean useSatelliteTime;
    private File kmlFile;
    private FileLock kmlLock;
    Kml10FileLogger(File kmlFile, boolean useSatelliteTime)
```

```
{
    this.useSatelliteTime = useSatelliteTime;
    this.kmlFile = kmlFile;
}
public void Write(Location loc) throws Exception
{
    try
    {
        Date now;
        if(useSatelliteTime)
        {
            now = new Date(loc.getTime());
        }
        else
        {
            now = new Date();
        }
        String dateTimeString = Utilities.GetIsoDateTime(now);
        if(!kmlFile.exists())
        {
            kmlFile.createNewFile();
            FileOutputStream initialWriter = new FileOutputStream(kmlFile, true);
            BufferedOutputStream initialOutput = new BufferedOutputStream(initialWriter);
            String initialXml = "<?xml version=\"1.0\"?>"
                    + "<kml xmlns=\"http://www.opengis.net/kml/2.2\"><Document>"
                    + "<Placemark><LineString><extrude>1</extrude><tessellate>1</tessellate>"
                    + "<altitudeMode>absolute</altitudeMode>"
                    + "<coordinates></coordinates></LineString></Placemark>"
                    + "</Document></kml>";
            initialOutput.write(initialXml.getBytes());
            initialOutput.flush();
            initialOutput.close();
        }
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document doc = builder.parse(kmlFile);
        NodeList coordinatesList = doc.getElementsByTagName("coordinates");
        if(coordinatesList.item(0) != null)
        {
            Node coordinates = coordinatesList.item(0);
            Node coordTextNode = coordinates.getFirstChild();
            if(coordTextNode == null)
            {
                coordTextNode = doc.createTextNode("");
                coordinates.appendChild(coordTextNode);
            }
            String coordText = coordinates.getFirstChild().getNodeValue();
            coordText = coordText + "\n" + String.valueOf(loc.getLongitude()) + ","
                    + String.valueOf(loc.getLatitude()) + "," + String.valueOf(loc.getAltitude());
            coordinates.removeChild(coordinates.getFirstChild());
            coordinates.appendChild(doc.createTextNode(coordText));
```

```
                }
                Node documentNode = doc.getElementsByTagName("Document").item(0);
                Node newPlacemark = doc.createElement("Placemark");
                Node timeStamp = doc.createElement("TimeStamp");
                Node whenNode = doc.createElement("when");
                Node whenNodeText = doc.createTextNode(dateTimeString);
                whenNode.appendChild(whenNodeText);
                timeStamp.appendChild(whenNode);
                newPlacemark.appendChild(timeStamp);
                Node newPoint = doc.createElement("Point");
                Node newCoords = doc.createElement("coordinates");
                newCoords.appendChild(doc.createTextNode(String.valueOf(loc.getLongitude()) + ","
                        + String.valueOf(loc.getLatitude()) + "," + String.valueOf(loc.getAltitude())));
                newPoint.appendChild(newCoords);
                newPlacemark.appendChild(newPoint);
                documentNode.appendChild(newPlacemark);
                String newFileContents = Utilities.GetStringFromNode(doc);
                synchronized(lock)
                {
                    FileOutputStream fos = new FileOutputStream(kmlFile, false);
                    fos.write(newFileContents.getBytes());
                    fos.close();
                }
            }
            catch(Exception e)
            {
                Utilities.LogError("Kml10FileLogger.Write", e);
                throw new Exception("Could not write to KML file");
            }
        }
        public void Annotate(String description) throws Exception
        {
            if(!kmlFile.exists())
            {
                return;
            }
            try
            {
                DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
                DocumentBuilder builder = factory.newDocumentBuilder();
                Document doc = builder.parse(kmlFile);
                NodeList placemarkList = doc.getElementsByTagName("Placemark");
                Node lastPlacemark = placemarkList.item(placemarkList.getLength() - 1);
                Node annotation = doc.createElement("name");
                annotation.appendChild(doc.createTextNode(description));
                lastPlacemark.appendChild(annotation);
                String newFileContents = Utilities.GetStringFromNode(doc);
                synchronized(lock)
                {
                    FileOutputStream fos = new FileOutputStream(kmlFile, false);
                    fos.write(newFileContents.getBytes());
```

```
                fos.close();
            }
        }
        catch(Exception e)
        {
            Utilities.LogError("Kml10FileLogger.Annotate", e);
            throw new Exception("Could not annotate KML file");
        }
    }
}
```

9.4　邮件分享提醒

知识点讲解：光盘:视频\知识点\第 9 章\邮件分享提醒.avi

在系统设置模块中，可以设置系统邮件来分享行走轨迹信息。其中邮件分享提醒界面如图 9-6 所示。

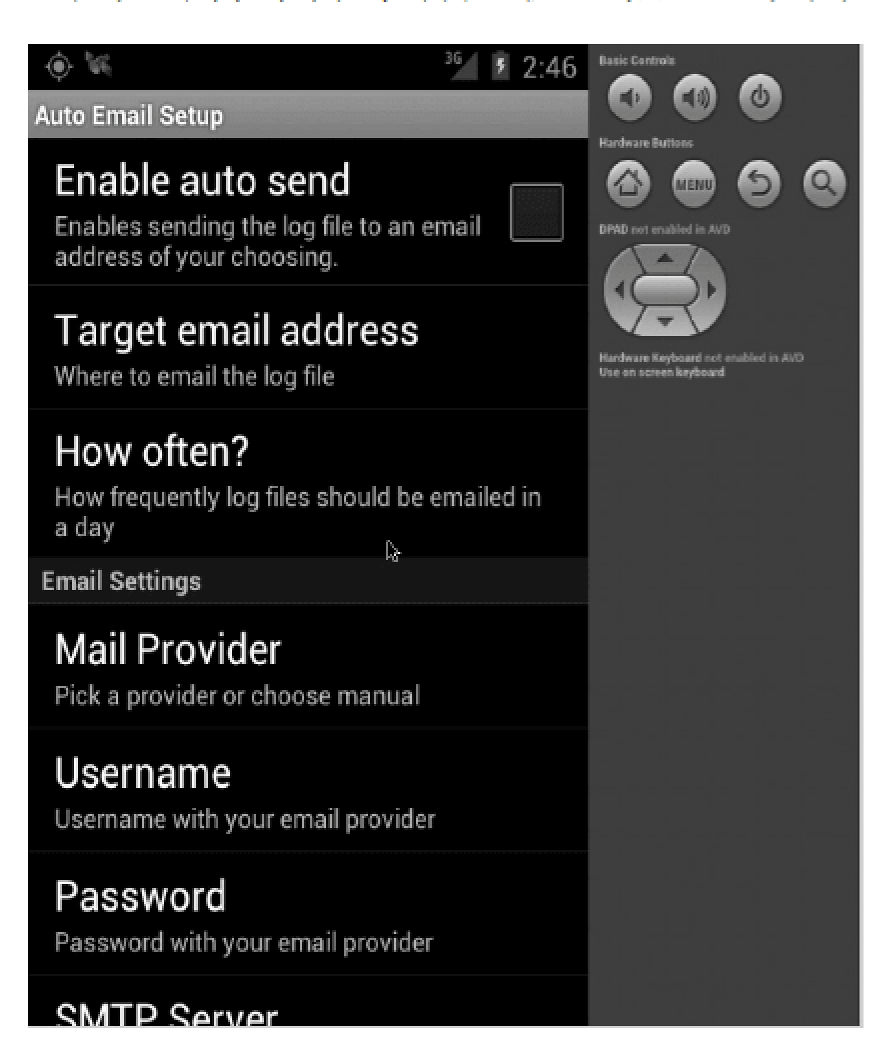


图 9-6　邮件分享提醒界面

9.4.1　基本邮箱设置

编写文件 AutoEmailActivity.java，功能是设置发送邮件的邮箱地址、密码、邮件服务器等信息，这样可以在使用时实现邮件自动发送功能。文件 AutoEmailActivity.java 的具体实现代码如下所示。

```
public class AutoEmailActivity extends PreferenceActivity implements
        OnPreferenceChangeListener, IMessageBoxCallback, IAutoSendHelper,
        OnPreferenceClickListener
{
    private final Handler handler   = new Handler();
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.autoemailsettings);
```

```
        CheckBoxPreference chkEnabled = (CheckBoxPreference) findPreference("autoemail_enabled");
        chkEnabled.setOnPreferenceChangeListener(this);
        ListPreference lstPresets = (ListPreference) findPreference("autoemail_preset");
        lstPresets.setOnPreferenceChangeListener(this);
        EditTextPreference txtSmtpServer = (EditTextPreference) findPreference("smtp_server");
        EditTextPreference txtSmtpPort = (EditTextPreference) findPreference("smtp_port");
        txtSmtpServer.setOnPreferenceChangeListener(this);
        txtSmtpPort.setOnPreferenceChangeListener(this);
        Preference testEmailPref = (Preference) findPreference("smtp_testemail");
        testEmailPref.setOnPreferenceClickListener(this);
    }
    public boolean onPreferenceClick(Preference preference)
    {
        if (!IsFormValid())
        {
            Utilities.MsgBox(getString(R.string.autoemail_invalid_form),
                    getString(R.string.autoemail_invalid_form_message),
                    AutoEmailActivity.this);
            return false;
        }
        Utilities.ShowProgress(this, getString(R.string.autoemail_sendingtest),
                    getString(R.string.please_wait));
        CheckBoxPreference chkUseSsl = (CheckBoxPreference) findPreference("smtp_ssl");
        EditTextPreference txtSmtpServer = (EditTextPreference) findPreference("smtp_server");
        EditTextPreference txtSmtpPort = (EditTextPreference) findPreference("smtp_port");
        EditTextPreference txtUsername = (EditTextPreference) findPreference("smtp_username");
        EditTextPreference txtPassword = (EditTextPreference) findPreference("smtp_password");
        EditTextPreference txtTarget = (EditTextPreference) findPreference("autoemail_target");
        AutoEmailHelper aeh = new AutoEmailHelper(null);
        aeh.SendTestEmail(txtSmtpServer.getText(), txtSmtpPort.getText(),
                txtUsername.getText(), txtPassword.getText(),
                chkUseSsl.isChecked(), txtTarget.getText(),
                AutoEmailActivity.this, AutoEmailActivity.this);
                return true;
    }
    private boolean IsFormValid()
    {
        CheckBoxPreference chkEnabled = (CheckBoxPreference) findPreference("autoemail_enabled");
        EditTextPreference txtSmtpServer = (EditTextPreference) findPreference("smtp_server");
        EditTextPreference txtSmtpPort = (EditTextPreference) findPreference("smtp_port");
        EditTextPreference txtUsername = (EditTextPreference) findPreference("smtp_username");
        EditTextPreference txtPassword = (EditTextPreference) findPreference("smtp_password");
        EditTextPreference txtTarget = (EditTextPreference) findPreference("autoemail_target");
        if (chkEnabled.isChecked())
        {
            if (txtSmtpServer.getText() != null
                    && txtSmtpServer.getText().length() > 0
                    && txtSmtpPort.getText() != null
                    && txtSmtpPort.getText().length() > 0
                    && txtUsername.getText() != null
                    && txtUsername.getText().length() > 0
```

```
                && txtPassword.getText() != null
                && txtPassword.getText().length() > 0
                && txtTarget.getText() != null
                && txtTarget.getText().length() > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    return true;
}
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    if (keyCode == KeyEvent.KEYCODE_BACK)
    {
        if (!IsFormValid())
        {
            Utilities.MsgBox(getString(R.string.autoemail_invalid_form),
                    getString(R.string.autoemail_invalid_form_message),
                    this);
            return false;
        }
        else
        {
            return super.onKeyDown(keyCode, event);
        }
    }
    else
    {
        return super.onKeyDown(keyCode, event);
    }
}
public void MessageBoxResult(int which)
{
    finish();
}
public boolean onPreferenceChange(Preference preference, Object newValue)
{
    if (preference.getKey().equals("autoemail_preset"))
    {
        int newPreset = Integer.valueOf(newValue.toString());
        switch (newPreset)
        {
            case 0:
                // Gmail
                SetSmtpValues("smtp.gmail.com", "465", true);
                break;
            case 1:
```

```
                    // Windows live mail
                    SetSmtpValues("smtp.live.com", "587", false);
                    break;
                case 2:
                    // Yahoo
                    SetSmtpValues("smtp.mail.yahoo.com", "465", true);
                    break;
                case 99:
                    // manual
                    break;
            }
        }
        return true;
    }
    private void SetSmtpValues(String server, String port, boolean useSsl)
    {
        SharedPreferences prefs = PreferenceManager
                .getDefaultSharedPreferences(getBaseContext());
        SharedPreferences.Editor editor = prefs.edit();
        EditTextPreference txtSmtpServer = (EditTextPreference) findPreference("smtp_server");
        EditTextPreference txtSmtpPort = (EditTextPreference) findPreference("smtp_port");
        CheckBoxPreference chkUseSsl = (CheckBoxPreference) findPreference("smtp_ssl");
        // Yahoo
        txtSmtpServer.setText(server);
        editor.putString("smtp_server", server);
        txtSmtpPort.setText(port);
        editor.putString("smtp_port", port);
        chkUseSsl.setChecked(useSsl);
        editor.putBoolean("smtp_ssl", useSsl);
        editor.commit();
    }
    String      testResults;
    public void OnRelay(boolean connectionSuccess, String message)
    {
        testResults = message;
        handler.post(showTestResults);
    }
    private final Runnable      showTestResults     = new Runnable()
                                                    {
                                                        public void run()
                                                        {
                                                            TestEmailResults();
                                                        }
                                                    };
    private void TestEmailResults()
    {
        Utilities.HideProgress();
        Utilities.MsgBox(getString(R.string.autoemail_testresult_title),
                testResults, this);
    }
}
```

9.4.2　实现邮件发送功能

编写文件 AutoEmailHelper.java，功能是使用邮件设置模块的邮箱来发送邮件信息，具体实现代码如下所示。

```
public class AutoEmailHelper implements IAutoSendHelper
{
	private GpsLoggingService	mainActivity;
	private boolean			forcedSend	= false;
	public AutoEmailHelper(GpsLoggingService activity)
	{
		this.mainActivity = activity;
	}
	public void SendLogFile(String currentFileName, boolean forcedSend)
	{
		this.forcedSend = forcedSend;
		Thread t = new Thread(new AutoSendHandler(currentFileName, this));
		t.start();
	}
	void SendTestEmail(String smtpServer, String smtpPort,
			String smtpUsername, String smtpPassword, boolean smtpUseSsl,
			String emailTarget, Activity callingActivity, IAutoSendHelper helper)
	{
		Thread t = new Thread(new TestEmailHandler(helper, smtpServer,
				smtpPort, smtpUsername, smtpPassword, smtpUseSsl, emailTarget));
		t.start();
	}
	public void OnRelay(boolean connectionSuccess, String errorMessage)
	{
		if (!connectionSuccess)
		{
			mainActivity.handler.post(mainActivity.updateResultsEmailSendError);
		}
		else
		{
			// This was a success
			Utilities.LogInfo("Email sent");
			if (!forcedSend)
			{
				Utilities.LogDebug("setEmailReadyToBeSent = false");
				Session.setEmailReadyToBeSent(false);
			}
		}
	}
}
class AutoSendHandler implements Runnable
{
	private String			currentFileName;
	private IAutoSendHelper	helper;
```

```
public AutoSendHandler(String currentFileName,      IAutoSendHelper helper)
{
        this.currentFileName = currentFileName;
        this.helper = helper;
}
public void run()
{
        File gpxFolder = new File(Environment.getExternalStorageDirectory(),
                        "GPSLogger");
        if (!gpxFolder.exists())
        {
                helper.OnRelay(true, null);
                return;
        }
        File gpxFile = new File(gpxFolder.getPath(), currentFileName + ".gpx");
        File kmlFile = new File(gpxFolder.getPath(), currentFileName + ".kml");
        File foundFile = null;
        if (kmlFile.exists())
        {
                foundFile = kmlFile;
        }
        if (gpxFile.exists())
        {
                foundFile = gpxFile;
        }
        if (foundFile == null)
        {
                helper.OnRelay(true, null);
                return;
        }
        String[ ] files = new String[ ]
        { foundFile.getAbsolutePath() };
        File zipFile = new File(gpxFolder.getPath(), currentFileName + ".zip");
        try
        {
                Utilities.LogInfo("Zipping file");
                ZipHelper zh = new ZipHelper(files, zipFile.getAbsolutePath());
                zh.Zip();
                Mail m = new Mail(AppSettings.getSmtpUsername(),
                                AppSettings.getSmtpPassword());
                String[ ] toArr =
                { AppSettings.getAutoEmailTarget() };
                m.setTo(toArr);
                m.setFrom(AppSettings.getSmtpUsername());
                m.setSubject("GPS Log file generated at "
                                + Utilities.GetReadableDateTime(new Date()) + " - "
                                + zipFile.getName());
                m.setBody(zipFile.getName());
                m.setPort(AppSettings.getSmtpPort());
                m.setSecurePort(AppSettings.getSmtpPort());
                m.setSmtpHost(AppSettings.getSmtpServer());
```

```
                m.setSsl(AppSettings.isSmtpSsl());
                m.addAttachment(zipFile.getAbsolutePath());
                Utilities.LogInfo("Sending email...");
                if (m.send())
                {
                    helper.OnRelay(true, "Email was sent successfully.");
                }
                else
                {
                    helper.OnRelay(false, "Email was not sent.");
                }
            }
            catch (Exception e)
            {
                helper.OnRelay(false, e.getMessage());
                Utilities.LogError("AutoSendHandler.run", e);
            }
        }
}

class TestEmailHandler implements Runnable
{
    String              smtpServer;
    String              smtpPort;
    String              smtpUsername;
    String              smtpPassword;
    boolean             smtpUseSsl;
    String              emailTarget;
    IAutoSendHelper     helper;
    public TestEmailHandler(IAutoSendHelper helper, String smtpServer,
                String smtpPort, String smtpUsername, String smtpPassword,
                boolean smtpUseSsl, String emailTarget)
    {
        this.smtpServer = smtpServer;
        this.smtpPort = smtpPort;
        this.smtpPassword = smtpPassword;
        this.smtpUsername = smtpUsername;
        this.smtpUseSsl = smtpUseSsl;
        this.emailTarget = emailTarget;
        this.helper = helper;
    }
    public void run()
    {
        try
        {
            Mail m = new Mail(smtpUsername, smtpPassword);
            String[ ] toArr =
            { emailTarget };
            m.setTo(toArr);
            m.setFrom(smtpUsername);
            m.setSubject("Test Email from GPSLogger at "
                    + Utilities.GetReadableDateTime(new Date()));
```

```
                m.setBody("Test Email from GPSLogger at "
                        + Utilities.GetReadableDateTime(new Date()));
                m.setPort(smtpPort);
                m.setSecurePort(smtpPort);
                m.setSmtpHost(smtpServer);
                m.setSsl(smtpUseSsl);
                m.setDebuggable(true);
                Utilities.LogInfo("Sending email...");
                if (m.send())
                {
                    helper.OnRelay(true, "Email was sent successfully.");
                }
                else
                {
                    helper.OnRelay(false, "Email was not sent.");
                }
            }
            catch (Exception e)
            {
                helper.OnRelay(false, e.getMessage());
                Utilities.LogError("AutoSendHandler.run", e);
            }
        }
}
```

9.5　上传 OSM 地图

知识点讲解：光盘:视频\知识点\第 9 章\上传 OSM 地图.avi

OSM 是 OpenStreetMap 的简称，这是一个网上地图协作计划，目的是创造一个内容自由且能让所有人编辑的世界地图。OSM 的地图由用户根据手提 GPS 装置、航空摄影照片、其他自由内容甚至单靠本地知识绘制。网站里的地图图像矢量数据皆以共享创意姓名表示，用相同方式分享 2.0 授权。OSM 网站的灵感来自维基百科等网站，经注册的用户可上传 GPS 路径及使用内置的编辑程式编辑数据。在上传地图信息之前，需要先获得授权标识。当单击 OpenStreetMap Preferences 按钮后会来到授权提示界面，在此界面会显示授权提示信息，如图 9-7 所示。

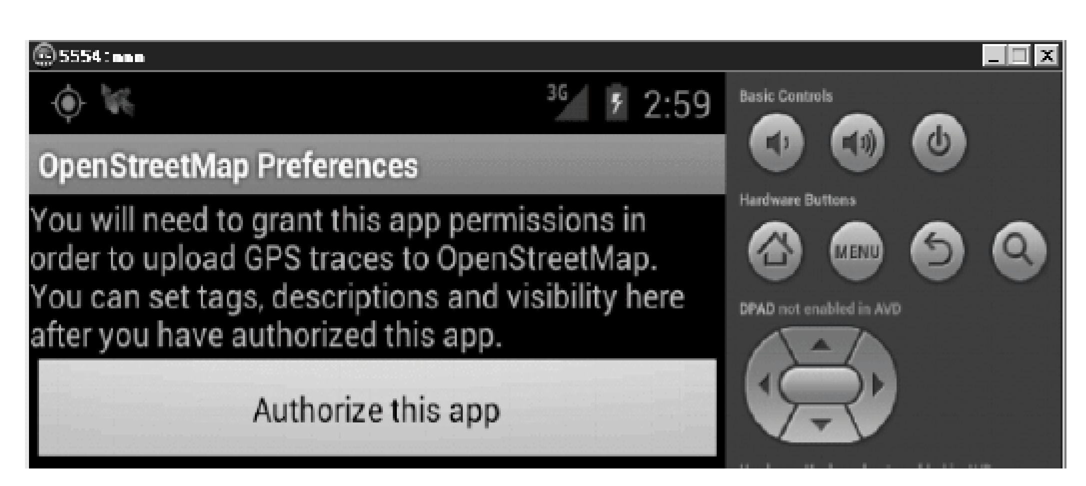


图 9-7　授权提示

9.5.1　授权提示布局文件

授权提示界面的布局文件是 osmauth.xml，具体实现代码如下所示。



```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
    <TableLayout android:id="@+id/TableOSM"
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:stretchColumns="1" android:background="#000000">
        <TableRow></TableRow>
        <TableRow>
        <TextView android:id="@+id/lblAuthorizeDescription" android:layout_height="wrap_content"
        android:text="@string/osm_lbl_authorize_description" android:layout_width="wrap_content"></TextView>
      </TableRow>
      <TableRow>
        <Button android:id="@+id/btnAuthorizeOSM"
                android:text="@string/osm_lbl_authorize"  android:layout_height="wrap_content"  android:layout_
width="wrap_content"/>
      </TableRow>
    </TableLayout>
</LinearLayout>
```

通过上述代码在屏幕中显示授权提示信息，并在屏幕下方显示了一个激活按钮。当单击激活按钮后会触发文件 OSMAuthorizationActivity.java，具体实现代码如下所示。

```
public class OSMAuthorizationActivity extends Activity implements
        OnClickListener
{
    private static OAuthProvider    provider;
    private static OAuthConsumer  consumer;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.osmauth);
        final Intent intent = getIntent();
        final Uri myURI = intent.getData();
        if (myURI != null && myURI.getQuery() != null
                && myURI.getQuery().length() > 0)
        {
            //User has returned! Read the verifier info from querystring
            String oAuthVerifier = myURI.getQueryParameter("oauth_verifier");
            try
            {
                SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(getBaseContext());

                if (provider == null)
                {
                    provider = Utilities.GetOSMAuthProvider(getBaseContext());
                }
                if (consumer == null)
                {
                    //In case consumer is null, re-initialize from stored values.
                    consumer = Utilities.GetOSMAuthConsumer(getBaseContext());
```

```
                }
                //Ask OpenStreetMap for the access token. This is the main event.
                provider.retrieveAccessToken(consumer, oAuthVerifier);

                String osmAccessToken = consumer.getToken();
                String osmAccessTokenSecret = consumer.getTokenSecret();

                //Save for use later.
                SharedPreferences.Editor editor = prefs.edit();
                editor.putString("osm_accesstoken", osmAccessToken);
                editor.putString("osm_accesstokensecret", osmAccessTokenSecret);
                editor.commit();

                //Now go away
                startActivity(new Intent(getBaseContext(), GpsMainActivity.class));
                finish();
            }
            catch (Exception e)
            {
                Utilities.LogError("OSMAuthorizationActivity.onCreate - user has returned", e);
                Utilities.MsgBox(getString(R.string.sorry), getString(R.string.osm_auth_error), this);
            }
        }
        Button authButton = (Button) findViewById(R.id.btnAuthorizeOSM);
        authButton.setOnClickListener(this);
    }
    public void onClick(View v)
    {
        try
        {
            //User clicks. Set the consumer and provider up.
            consumer = Utilities.GetOSMAuthConsumer(getBaseContext());
            provider = Utilities.GetOSMAuthProvider(getBaseContext());
            String authUrl;
            //Get the request token and request token secret
            authUrl = provider.retrieveRequestToken(consumer, OAuth.OUT_OF_BAND);
            //Save for later
            SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(getBaseContext());
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("osm_requesttoken", consumer.getToken());
            editor.putString("osm_requesttokensecret",consumer.getTokenSecret());
            editor.commit();
            //Open browser, send user to OpenStreetMap.org
            Uri uri = Uri.parse(authUrl);
            Intent intent = new Intent(Intent.ACTION_VIEW, uri);
            startActivity(intent);
        }
        catch (Exception e)
        {
            Utilities.LogError("OSMAuthorizationActivity.onClick", e);
            Utilities.MsgBox(getString(R.string.sorry), getString(R.string.osm_auth_error), this);
```

```
            }
        }
}
```

9.5.2　实现文件上传

编写文件 OSMHelper.java，功能是在获取权限后上传 OpenStreetMap 轨迹文件，具体实现代码如下所示。

```
public class OSMHelper implements IOsmHelper
{
        private GpsMainActivity mainActivity;
        public OSMHelper(GpsMainActivity activity)
        {
                this.mainActivity = activity;
        }

        public void UploadGpsTrace(String fileName)
        {
                File gpxFolder = new File(Environment.getExternalStorageDirectory(), "GPSLogger");
                File chosenFile = new File(gpxFolder, fileName);
                OAuthConsumer consumer = Utilities.GetOSMAuthConsumer(mainActivity.getBaseContext());
                String gpsTraceUrl = mainActivity.getString(R.string.osm_gpstrace_url);

                SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(mainActivity.getBaseContext());
                String description = prefs.getString("osm_description", "");
                String tags = prefs.getString("osm_tags", "");
                String visibility = prefs.getString("osm_visibility", "private");

                Thread t = new Thread(new OsmUploadHandler(this, consumer, gpsTraceUrl, chosenFile, description,
tags, visibility));
                t.start();
        }
        public void OnComplete()
        {
                mainActivity.handler.post(mainActivity.updateOsmUpload);
        }
        private class OsmUploadHandler implements Runnable
        {
                OAuthConsumer consumer;
                String gpsTraceUrl;
                File chosenFile;
                String description;
                String tags;
                String visibility;
                IOsmHelper helper;

                public OsmUploadHandler(IOsmHelper helper, OAuthConsumer consumer, String gpsTraceUrl, File
chosenFile, String description, String tags, String visibility)
                {
                        this.consumer = consumer;
```

```
                this.gpsTraceUrl = gpsTraceUrl;
                this.chosenFile = chosenFile;
                this.description = description;
                this.tags = tags;
                this.visibility = visibility;
                this.helper = helper;
            }

            public void run()
            {
                try
                {
                    HttpPost request = new HttpPost(gpsTraceUrl);

                    consumer.sign(request);
                    MultipartEntity entity = new MultipartEntity(HttpMultipartMode.BROWSER_COMPATIBLE);

                    FileBody gpxBody = new FileBody(chosenFile);
                    entity.addPart("file", gpxBody);
                    if(description == null || description.length() <= 0)
                    {
                        description = "GPSLogger for Android";
                    }

                    entity.addPart("description", new StringBody(description));
                    entity.addPart("tags", new StringBody(tags));
                    entity.addPart("visibility", new StringBody(visibility));

                    request.setEntity(entity);
                    DefaultHttpClient httpClient = new DefaultHttpClient();
                    HttpResponse response = httpClient.execute(request);
                    int statusCode = response.getStatusLine().getStatusCode();
                    Utilities.LogDebug("OSM Upload - " + String.valueOf(statusCode));
                    helper.OnComplete();

                }
                catch(Exception e)
                {
                        Utilities.LogError("OsmUploadHelper.run", e);
                }
            }
        }
}
interface IOsmHelper
{
        public void OnComplete();
}
```

到此为止，本章实例的主要模块介绍完毕。为了节省本书的篇幅，有关本实例其余模块的具体实现过程，请读者参阅本书附带光盘的内容。

第 10 章　智能家居系统

智能家居是一个居住环境，又称智能住宅。通俗地说，它是融合了自动化控制系统、计算机网络系统、无限传感网络系统和网络通信技术于一体的网络化、智能化的家居控制系统。本章将通过一个综合实例的实现过程，详细讲解在 Android 系统中开发一个智能家居系统的方法。

10.1　需 求 分 析

知识点讲解：光盘:视频\知识点\第 10 章\需求分析.avi

智能家居将让用户以更便捷的方式来管理家庭设备，比如，通过触摸屏、无线遥控器、电话、互联网或者语音识别控制家用设备，更可以执行场景操作，使多个设备形成联动；另外，智能家居内的各种设备相互间可以通信，不需要用户指挥也能根据不同的状态互动运行，从而给用户带来最大程度的高效、便利、舒适与安全。在本节的内容中，将详细讲解本系统的需求分析。

10.1.1　背景介绍

家居智能化在我国的应用已经有一段时间了，但是目前大多数的智能家居系统仍然只适用于别墅、洋房、公寓等高级住房。而移动电话的应用在我国却非常普及，所以手机智能家居系统软件将成为智能家居系统中的主流产品。

对于智能家居产品，第一印象便是便捷，通过一个小小的手机，便可随时掌握、控制家里的所有常用家电设备，包括：灯光、窗帘、电器、空调和地暖等，甚至天气预报，室内温湿度显示等，成为未来理想智能家居的必需品。随着各种基于 3G 和 WiFi 功能的智能产品逐步应用于人们的生活中，方便直观触摸操作的移动触摸智能控制终端诸如 Android、iPhone、iPad 等，必将成为智能家居未来的发展趋势。传统的智能家居系统包含的主要子系统有：家居布线系统、家庭网络系统、智能家居（中央）控制管理系统、家居照明控制系统、家庭安防系统、背景音乐系统、家庭影院与多媒体系统、家庭环境控制系统等 8 大系统。其中，智能家居（中央）控制管理系统、家居照明控制系统、家庭安防系统是必备系统，家居布线系统、家庭网络系统、背景音乐系统、家庭影院与多媒体系统、家庭环境控制系统为可选系统。

通常会认为智能家居会带来生活品质的提升，其实物联网智能家居正在改变这些观点，最显著的变化就是实用、方便、易整合。每一个家庭中都存在各种电器，不管是号称智能的冰箱、空调还是传统的电灯、电视，一直以来由于标准不一都是独立工作的，从系统的角度来看，它们都是零碎的、混乱的、无序的，并不是一个有机的、可组织的整体，这些杂乱无章的电器所消耗的时间成本、管理成本、控制成本通常都是很高的，作为家庭的主人对其进行智能化管理非常必要。

10.1.2 传感技术的推动

传感器技术是近年来在科技界兴起的一大热点，这项技术是全球第一个利用物联网来控制灯饰及电子电器产品（ZigBee 标准产品），并将其作为智能家居主流产品走向了商业化。ZigBee 最初预计的应用领域主要包括消费电子、能源管理、卫生保健、家庭自动化、建筑自动化和工业自动化。随着物联网的兴起，ZigBee 又获得了新的应用机会。物联网的网络边缘应用最多的就是传感器或控制单元，这些是构成物联网最基础、最核心、最广泛的单元，而 ZigBee 能够在数千个微小的传感传动单元之间相互协调实现通信，并且这些单元只需要很少的能量，以接力的方式通过无线电波将数据从一个网络节点传到另一个节点，所以它的通信效率非常高。这种技术低功耗、抗干扰、高可靠、易组网、易扩容、易使用、易维护、便于快速大规模部署等特点顺应了物联网发展的要求和趋势。目前来看，物联网和 ZigBee 技术在智能家居、工业监测和健康保健等方面的应用有很大的融合性。

值得注意的是，物联网的兴起将会给 ZigBee 带来广阔的市场空间。因为物联网的目的是要将各种信息传感传动单元与互联网结合起来从而形成一个巨大的网络，在这个巨大网络中，传感传动单元与通信网络之间需要数据的传输，而相对其他无线技术而言，ZigBee 以其在投资、建设、维护等方面的优势，必将在物联网型智能家居领域获得更广泛的应用。物联传感控制规格遂成为当今家庭智能家居自动化控制规格的主要领导者。

10.1.3 Android 与智能家居的紧密联系

2011 年 5 月 10 日，Google 在 Moscone West 会议中心举行的 Google I/O 大会上首次宣布了 Android@Home 计划。究竟什么是 Android@Home？就是通过 Android 设备来完全控制例如台灯、闹钟、洗碗机等家电产品，实现居住环境的完全 Android 化。

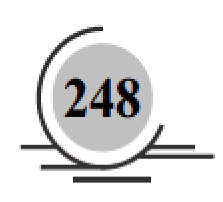
在 2011 年举办的谷歌 I/O 大会展示当中，谷歌为我们演示了通过摩托罗拉 XOOM 平板上对 4 台台灯进行控制。用户只需对平板上的 4 个开关进行虚拟化操作，对应的台灯就会立马作出相应反应。下一个例子，谷歌向我们展示了唤醒懒人的好方法：通过 Android@Home 连接在一起的闹铃系统。每当闹铃响起之际，家中音响的音量、灯光的亮度也会随之递增，面对如此先进的闹铃系统，相信再能睡的人也会被其准时唤醒。

现在，Android@Home 计划又有了新的进展，正在有计划建立一个连接家中所有 Android 设备的云端自动化平台。因为后 PC 时代的一部分组成是云，所以 Google 正在努力地成为一个云服务提供商。随着 Google 对 Android@Home 计划的投入，越来越多的新技术会被应用到智能家居中，这也将引爆新一轮的智能家居创新热潮。

10.2 系统功能模块介绍

知识点讲解：光盘:视频\知识点\第 10 章\系统功能模块介绍.avi

本章智能家居系统的功能是，在 Android 系统中开发一个智能家居系统客户端 App，通过这个 App 可以实现照明控制、温度控制、查看天气和系统设置功能。本章智能家居系统的构成模块结构如图 10-1 所示。



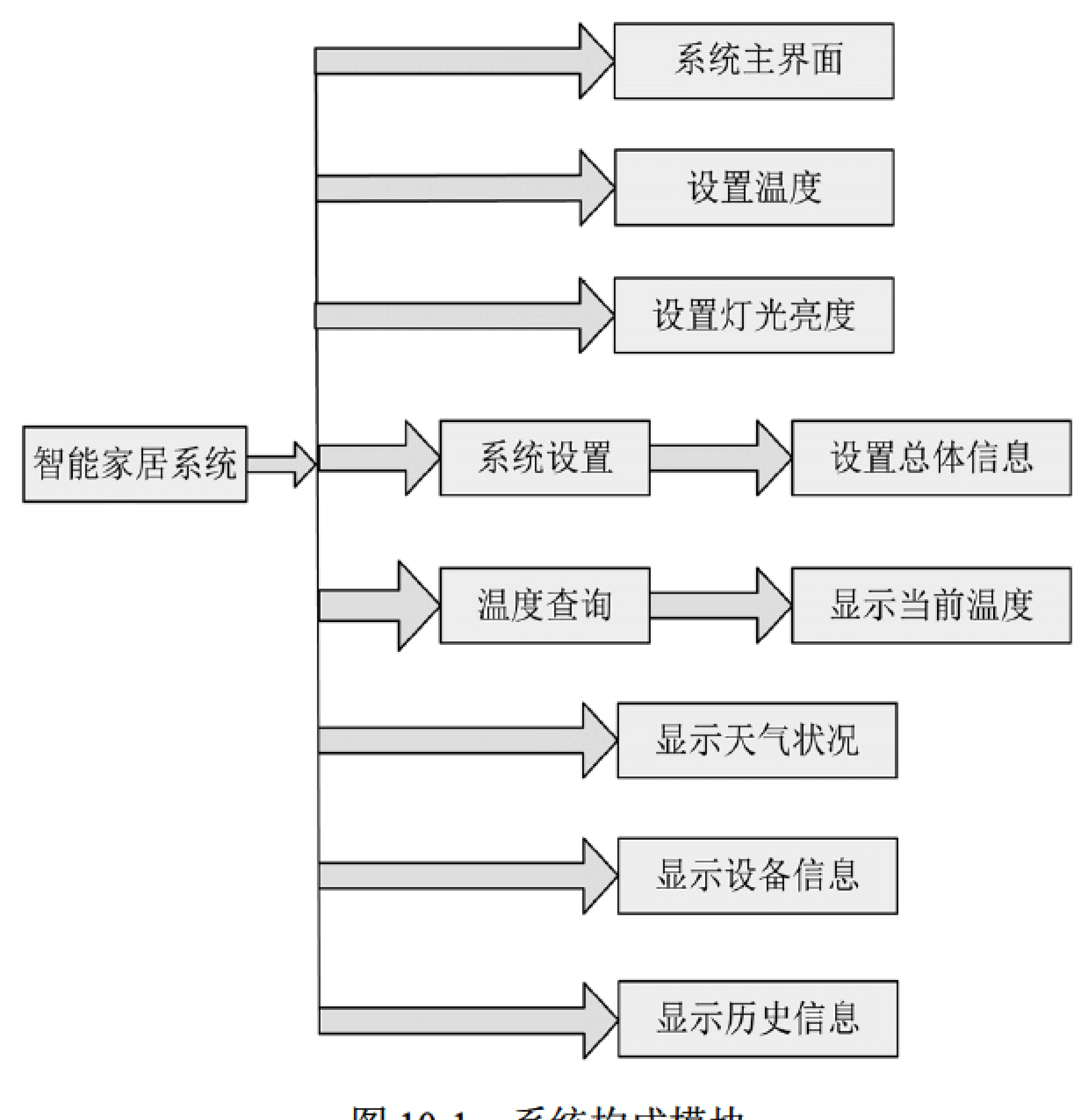


图 10-1　系统构成模块

10.3　系统主界面

知识点讲解：光盘:视频\知识点\第 10 章\系统主界面.avi

系统主界面是运行程序后首先呈现在用户面前的界面。在本节的内容中，将详细讲解实现本章智能家居系统主界面的具体流程。

10.3.1　实现布局文件

通过分析 AndroidManifest.xml 清单文件可知，本系统主界面的布局文件是 main.xml，功能是通过 GridView 控件显示系统控制选项。文件 main.xml 的具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />

    <GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:numColumns="auto_fit"
```

```
        android:verticalSpacing="10dp"
        android:horizontalSpacing="10.dp"
        android:columnWidth="90dp"
        android:stretchMode="columnWidth"
        android:gravity="center"
        />

</LinearLayout>
```

10.3.2 实现程序文件

系统主界面的程序文件是 ClientStart.java，功能是加载布局文件 main.xml，然后在 GridView 控件中显示系统控制选项：温度、电器控制、预案管理。文件 ClientStart.java 的具体实现代码如下所示。

```
public class ClientStart extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        GridView gridView=(GridView)findViewById(R.id.gridview);
        ArrayList<HashMap<String, Object>> lstImageItem=
          new ArrayList<HashMap<String, Object>>();

        HashMap<String, Object> map0=new HashMap<String,Object>();
        map0.put("Image", R.drawable.app_icon);
        map0.put("Text", "温度");
        lstImageItem.add(map0);

        HashMap<String, Object> map1=new HashMap<String,Object>();
        map1.put("Image", R.drawable.sln);
        map1.put("Text", "电器控制");
        lstImageItem.add(map1);

        HashMap<String, Object> map2=new HashMap<String,Object>();
        map2.put("Image", R.drawable.open);
        map2.put("Text", "预案管理");
        lstImageItem.add(map2);

        HashMap<String, Object> map3=new HashMap<String,Object>();
        map3.put("Image", R.drawable.close);
        map3.put("Text", "4");
        lstImageItem.add(map2);

        SimpleAdapter saImageItems=new SimpleAdapter(this,lstImageItem,R.layout.function,new String[ ]{"Image",
"Text"}, new int[ ]{R.id.ItemImage,R.id.ItemText});

        gridView.setAdapter(saImageItems);

        gridView.setOnItemClickListener(new ItemClickListener());
```

```
        }
    class ItemClickListener implements OnItemClickListener{
     public void onItemClick(AdapterView<?> arg0,View arg1,int arg2,long arg3)
     {
            Intent intent;
            switch (arg2)
            {
            case 0:
                intent= new Intent (ClientStart.this,SystemSet.class);
                startActivity(intent);
                break;
            case 1:
                intent = new Intent (ClientStart.this,DeviceControl.class);
                startActivity(intent);
                break;
            case 2:
                intent = new Intent (ClientStart.this,Weather01.class);
                startActivity(intent);
                break;
            case 3:
                break;
                default:
                break;
            }
     }
  }
}
```

系统主界面的执行效果如图 10-2 所示。



图 10-2　系统主界面

10.4　系 统 设 置

知识点讲解：光盘:视频\知识点\第 10 章\系统设置.avi

系统设置模块的功能是设置当前使用者的系统信息，主要包括服务器电话、控制方式、短信服务、城市名称、更新频率、开启预案、服务器地址和端口信息等。在本节的内容中，将详细讲解实现本章系统设置模块的具体流程。

10.4.1 总体配置

编写文件 SystemConst.java，功能是实现系统的总体变量配置，这些变量包括服务器电话、控制方式、短信服务、城市名称、更新频率、开启预案、服务器地址和端口信息等。文件 SystemConst.java 的具体实现代码如下所示。

```
public class SystemConst {
  public static final String DB_NAME ="SmartHome_Client.db";
  public static final int DB_VERSION = 1;

  public static final String DB_TABLE_CONFIG = "system_config";
  public static final String KEY_ID="id";
  public static final String KEY_SERVER_TEL = "server_tel";
  public static final String KEY_CONTROL_METHOD = "control_method";
  public static final String KEY_CITY_NAME = "city_name";
  public static final String KEY_REFRESH_SPEED = "refresh_speed";
  public static final String KEY_WEATHER_SERVICE = "weather_service";
  public static final String KEY_LOCATION_SERVICE = "location_service";
  public static final String KEY_START_TIME = "start_time";
  public static final String KEY_SERVER_IP = "server_ip";
  public static final String KEY_SERVER_COM = "server_com";
}
```

10.4.2 系统总体配置

编写系统设置界面的布局文件 system_set.xml，功能是通过文本控件、文本框控件和单选按钮控件来显示当前系统的设置信息。文件 system_set.xml 的具体实现代码如下所示。

```
<AbsoluteLayout
android:id="@+id/widget0"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>

<EditText
    android:id="@+id/server_tel"
    android:layout_width="200px"
    android:layout_height="27dp"
    android:layout_x="111px"
    android:layout_y="9px"
    android:textSize="12sp" >

</EditText>
<TextView
android:id="@+id/widget37"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:text="&#26381;&#21153;&#22120;&#30005;&#35805;&#65306;"
android:layout_x="6px"
android:layout_y="21px"
>
</TextView>
<TextView
android:id="@+id/widget38"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#30005;&#22120;&#25511;&#21046;&#26041;&#24335;&#65306;"
android:layout_x="5px"
android:layout_y="73px"
>
</TextView>
<RadioGroup
android:id="@+id/radiogroup"
android:layout_width="147px"
android:layout_height="14px"
android:orientation="horizontal"
android:layout_x="115px"
android:layout_y="64px"
>
</RadioGroup>
<TextView
android:id="@+id/location_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#25552;&#20379;&#23450;&#20301;&#30701;&#20449;&#26381;&#21153;&#65306;"
android:layout_x="4px"
android:layout_y="133px"
>
</TextView>
<TextView
android:id="@+id/city_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#22478;&#24066;&#21517;&#31216;&#65288;&#25340;&#38899;&#65289;&#65306;"
android:layout_x="4px"
android:layout_y="182px"
>
</TextView>

<EditText
    android:id="@+id/edit_city"
    android:layout_width="147dp"
    android:layout_height="30dp"
    android:layout_x="140dp"
    android:layout_y="171px"
    android:text="Beijing"
    android:textSize="12sp" >
```

```
</EditText>
<TextView
android:id="@+id/widget46"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#26356;&#26032;&#39057;&#29575;&#65306;"
android:layout_x="6px"
android:layout_y="226px"
>
</TextView>
<TextView
android:id="@+id/widget49"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#24320;&#21551;&#22825;&#27668;&#39044;&#26696;&#25552;&#31034;&#65306;"
android:layout_x="6px"
android:layout_y="265px"
>
</TextView>
<TextView
android:id="@+id/widget52"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#39044;&#26696;&#24320;&#21551;&#26102;&#21051;&#65306;"
android:layout_x="6px"
android:layout_y="307px"
>
</TextView>
<TextView
android:id="@+id/widget56"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Server_IP:"
android:layout_x="6px"
android:layout_y="346px"
>
</TextView>

<TextView
    android:id="@+id/widget48"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="78dp"
    android:layout_y="153dp"
    android:text="秒/次" />

<EditText
    android:id="@+id/frequence"
    android:layout_width="139px"
```

```
    android:layout_height="28dp"
    android:layout_x="118dp"
    android:layout_y="148dp"
    android:ems="10"
    android:text="60"
    android:textSize="12sp" />

<Button
    android:id="@+id/apply"
    android:layout_width="122dp"
    android:layout_height="wrap_content"
    android:layout_x="11dp"
    android:layout_y="340dp"
    android:text="应用系统设置"
    android:textStyle="bold" />

<TextView
    android:id="@+id/widget58"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="153dp"
    android:layout_y="231dp"
    android:text="端口：" />

<EditText
    android:id="@+id/edit_ip"
    android:layout_width="83dp"
    android:layout_height="29dp"
    android:layout_x="70dp"
    android:layout_y="228dp"
    android:ems="10"
    android:text="127.0.0.1"
    android:textSize="12sp" />

<RadioButton
    android:id="@+id/radiobutton_blue"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="119dp"
    android:layout_y="40dp"
    android:text="蓝牙" />

<RadioButton
    android:id="@+id/radiobutton_sms"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="210dp"
    android:layout_y="40dp"
    android:checked="true"
    android:text="短信" />
```

```
<CheckBox
    android:id="@+id/checkbox01"
    android:layout_width="82dp"
    android:layout_height="28dp"
    android:layout_x="130dp"
    android:layout_y="86dp"
    android:checked="true"
    android:text="是" />

<EditText
    android:id="@+id/edit_com"
    android:layout_width="94dp"
    android:layout_height="30dp"
    android:layout_x="196dp"
    android:layout_y="232dp"
    android:ems="10"
    android:text="5432"
    android:textSize="12sp" />

<CheckBox
    android:id="@+id/checkbox_plan"
    android:layout_width="wrap_content"
    android:layout_height="30dp"
    android:layout_x="118dp"
    android:layout_y="170dp"
    android:text="是" />

<EditText
    android:id="@+id/planstart_time"
    android:layout_width="84dp"
    android:layout_height="34dp"
    android:layout_x="109dp"
    android:layout_y="196dp"
    android:ems="10"
    android:text="09:00"
    android:textSize="12sp" />

<Button
    android:id="@+id/reset"
    android:layout_width="124dp"
    android:layout_height="wrap_content"
    android:layout_x="148dp"
    android:layout_y="340dp"
    android:text="取消系统设置"
    android:textStyle="bold" />

</AbsoluteLayout>
```

编写 SystemSet.java 文件，功能是获取设置界面文本框中的值和单选按钮的值，并将这些值保存在



系统数据库中以实现系统设置功能。文件 SystemSet.java 的具体实现代码如下所示。

```
public class SystemSet extends Activity{

    private EditText serverTel;
    private EditText serverIp;
    private EditText serverCom;
    private RadioButton smsMethodView;
    private RadioButton blueToothMethodView;
    private EditText cityNameView;
    private EditText refreshSpeedView;
    private CheckBox weatherServiceView;
    private CheckBox locationServiceView;
    private EditText startTimeView;
    private Button applyBtn;
    private Button resetBtn;
    public static DBAdapter dbAdapter;


     public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.system_set);

        serverTel = (EditText)findViewById(R.id.server_tel);
        serverIp = (EditText)findViewById(R.id.edit_ip);
        serverCom = (EditText)findViewById(R.id.edit_com);
        smsMethodView=(RadioButton)findViewById(R.id.radiobutton_sms);
        blueToothMethodView=(RadioButton)findViewById(R.id.radiobutton_blue);
        cityNameView=(EditText)findViewById(R.id.edit_city);
        refreshSpeedView=(EditText)findViewById(R.id.frequence);
        weatherServiceView=(CheckBox)findViewById(R.id.checkbox_plan);
        locationServiceView=(CheckBox)findViewById(R.id.checkbox01);
        startTimeView=(EditText)findViewById(R.id.planstart_time);
        applyBtn =(Button)findViewById(R.id.apply);
        resetBtn = (Button)findViewById(R.id.reset);

        dbAdapter = new DBAdapter(this);
        dbAdapter.open();
        dbAdapter.LoadConfig();

        applyBtn.setOnClickListener(new View.OnClickListener(){
          public void onClick(View v) {
                SetNewData();
                UpdateUi();
          }
        });

        resetBtn.setOnClickListener(new View.OnClickListener(){
          public void onClick(View v){
                dbAdapter.LoadConfig();
```

```
                UpdateUi();
            }
        });

        UpdateUi();
    }

    private void SetNewData(){
            Config.ServerTel = serverTel.getText().toString();
            if(smsMethodView.isChecked()==true)
                Config.ControlMethod = "sms";
            else {
                Config.ControlMethod = "bluetooth";
            }
            if (blueToothMethodView.isChecked()==true)
                Config.ControlMethod = "bluetooth";
            else {
                Config.ControlMethod = "sms";
            }
            Config.CityName = cityNameView.getText().toString();
            Config.RefreshSpeed = refreshSpeedView.getText().toString();

            if (weatherServiceView.isChecked() == true)
                Config.WeatherService = "true";
            else {
                Config.WeatherService = "false";
            }
            if (locationServiceView.isChecked()==true)
                Config.LocationService = "true";
            else {
                Config.LocationService = "false";
            }
            Config.StartTime = startTimeView.getText().toString();
            Config.ServerIP = serverIp.getText().toString();
            Config.ServerCom = serverCom.getText().toString();

            dbAdapter.SaveConfig();
    }

    private void UpdateUi(){
            serverTel.setText(Config.ServerTel);

            if(Config.ControlMethod.equals("sms")==true){
                //smsMethodView.setChecked(false);
                //blueToothMethodView.setChecked(true);
            }
            locationServiceView.setChecked(Config.LocationService.equals("true")?true:false);
            cityNameView.setText(Config.CityName);
            refreshSpeedView.setText(Config.RefreshSpeed);
            weatherServiceView.setChecked(Config.WeatherService.equals("true")?true:false);
```

```
            startTimeView.setText(Config.StartTime);
            serverIp.setText(Config.ServerIP);
            serverCom.setText(Config.ServerCom);
        }
}
```

10.4.3　构建数据库

本系统中的设置信息是被保存在 SQLite 数据库中的，本实例通过文件 DBOpenHelper.java 来构建数据库的参数值，具体实现代码如下所示。

```
public class DBOpenHelper extends SQLiteOpenHelper{

    public DBOpenHelper(Context context,String name,
                CursorFactory factory,int version){
        super(context,name,factory,version);
    }
    //创建“系统设置”表结构
    private static final String DB_CREATE_CONFIG = "create table "
        + SystemConst.DB_TABLE_CONFIG + " (" + SystemConst.KEY_ID
        + " integer primary key autoincrement, " + SystemConst.KEY_SERVER_TEL
        + " text not null, " + SystemConst.KEY_CONTROL_METHOD + " text, "
        + SystemConst.KEY_CITY_NAME + " text, " + SystemConst.KEY_REFRESH_SPEED + " text,"
        + SystemConst.KEY_WEATHER_SERVICE + " text, " + SystemConst.KEY_LOCATION_SERVICE  +  " text, "
        + SystemConst.KEY_START_TIME + " text, "+ SystemConst.KEY_SERVER_IP +" text, "+SystemConst.
KEY_SERVER_COM+" text);";
    //创建“天气预案”表
    private final String DB_CREATE_RESERVEPLAN = "create table Reserve_Plan (_id integer primary key
autoincrement,"
        + "weather varchar(100), temperature varchar(100),"
        + "solution varchar(300));";
    @Override
    public void onCreate(SQLiteDatabase _db) {
        // TODO Auto-generated method stub
        //创建“系统设置”和“天气预案”表
        _db.execSQL(DB_CREATE_CONFIG);
        _db.execSQL(DB_CREATE_RESERVEPLAN);
        //系统设置默认数据加载
        Config.LoadDefaultConfig();
        //系统设置默认数据插入数据库表中
        ContentValues newValues = new ContentValues();
        newValues.put(SystemConst.KEY_SERVER_TEL, Config.ServerTel);
        newValues.put(SystemConst.KEY_CONTROL_METHOD, Config.ControlMethod);
        newValues.put(SystemConst.KEY_CITY_NAME,Config.CityName);
        newValues.put(SystemConst.KEY_REFRESH_SPEED, Config.RefreshSpeed);
        newValues.put(SystemConst.KEY_WEATHER_SERVICE, Config.WeatherService);
        newValues.put(SystemConst.KEY_LOCATION_SERVICE, Config.LocationService);
        newValues.put(SystemConst.KEY_START_TIME, Config.StartTime);
        newValues.put(SystemConst.KEY_SERVER_IP, Config.ServerIP);
        newValues.put(SystemConst.KEY_SERVER_COM, Config.ServerCom);
        _db.insert(DB_CREATE_CONFIG, null, newValues);
```

```
    }

    @Override
    public void onUpgrade(SQLiteDatabase _db, int _oldVersion, int _newVersion) {
        // TODO Auto-generated method stub
        _db.execSQL("DROP TABLE IF EXISTS"+SystemConst.DB_TABLE_CONFIG);
        _db.execSQL("DROP TABLE IF EXISTS Reserve_Plan");
        onCreate(_db);
    }

}
```

编写 DBAdapter.java 文件，功能是根据用户的设置信息构建或更新数据库数据，具体实现代码如下所示。

```
public class DBAdapter {
    private SQLiteDatabase db;
    private final Context context;
    private DBOpenHelper dbOpenHelper;

    public DBAdapter (Context _context){
        context = _context;
    }

    public void open()throws SQLiteException{
        dbOpenHelper = new DBOpenHelper(context,SystemConst.DB_NAME,null,SystemConst.DB_VERSION);
        try{
            db = dbOpenHelper.getWritableDatabase();
        }catch(SQLiteException ex){
            db = dbOpenHelper.getReadableDatabase();
        }
    }

    //从数据库查找系统设置表，加载数据到系统常量
    public void LoadConfig(){
        Cursor result =  db.query(SystemConst.DB_TABLE_CONFIG, new String[ ] { SystemConst.KEY_ID,
SystemConst.KEY_SERVER_TEL, SystemConst.KEY_CONTROL_METHOD,
                            SystemConst.KEY_CITY_NAME, SystemConst.KEY_REFRESH_SPEED,
                            SystemConst.KEY_WEATHER_SERVICE,  SystemConst.KEY_LOCATION_SERVICE,
SystemConst.KEY_START_TIME,SystemConst.KEY_SERVER_IP,SystemConst.KEY_SERVER_COM},
                            SystemConst.KEY_ID + "=1", null, null, null, null);
                if (result.getCount() == 0 || !result.moveToFirst()){
                        return;
                    }
                Config.ServerTel = result.getString(result.getColumnIndex(SystemConst.KEY_SERVER_TEL));
                Config.ControlMethod = result.getString(result.getColumnIndex(SystemConst.KEY_CONTROL_
METHOD));
                Config.CityName = result.getString(result.getColumnIndex(SystemConst.KEY_CITY_NAME));
                Config.RefreshSpeed =  result.getString(result.getColumnIndex(SystemConst.KEY_REFRESH_
SPEED));
                Config.WeatherService = result.getString(result.getColumnIndex(SystemConst.KEY_WEATHER_
SERVICE));
```

```
                Config.LocationService =   result.getString(result.getColumnIndex(SystemConst.KEY_
LOCATION_SERVICE));
                Config.StartTime = result.getString(result.getColumnIndex(SystemConst.KEY_START_TIME));
                Config.ServerIP =result.getString(result.getColumnIndex(SystemConst.KEY_SERVER_IP));
                Config.ServerCom =result.getString(result.getColumnIndex(SystemConst.KEY_SERVER_COM));
                Toast.makeText(context, "系统设置读取成功", Toast.LENGTH_SHORT).show();

        }

        public void SaveConfig(){
            ContentValues updateValues =   new ContentValues();
            updateValues.put(SystemConst.KEY_CITY_NAME, Config.CityName);
            updateValues.put(SystemConst.KEY_CONTROL_METHOD, Config.ControlMethod);
            updateValues.put(SystemConst.KEY_LOCATION_SERVICE, Config.LocationService);
            updateValues.put(SystemConst.KEY_REFRESH_SPEED, Config.RefreshSpeed);
            updateValues.put(SystemConst.KEY_SERVER_COM, Config.ServerCom);
            updateValues.put(SystemConst.KEY_SERVER_IP, Config.ServerIP);
            updateValues.put(SystemConst.KEY_SERVER_TEL, Config.ServerTel);
            updateValues.put(SystemConst.KEY_START_TIME, Config.StartTime);
            updateValues.put(SystemConst.KEY_WEATHER_SERVICE, Config.WeatherService);
            db.update(SystemConst.DB_TABLE_CONFIG, updateValues,
                        SystemConst.KEY_ID+"=1", null);
            Toast.makeText(context, "系统设置保存成功", Toast.LENGTH_SHORT).show();
        }

        public void close(){
            if (db!=null){
                db.close();
                db = null;
            }
        }
}
```

10.5　电器控制模块

知识点讲解：光盘:视频\知识点\第 10 章\电器控制模块.avi

在系统主界面中触摸按下“电器控制”图标后会来到如图 10-3 所示的界面。由此可见，通过电器控制模块可以控制家居的温度、电灯和电扇。在本节的内容中，将详细讲解本章电器控制模块的具体实现流程。



图 10-3　电器控制界面

10.5.1　电器控制主界面

编写布局文件 devicecontrol.xml 实现电器控制模块的主界面，在主界面中通过 ListView 控件默认加载显示温度、电灯和电扇 3 个列表选项。文件 devicecontrol.xml 的具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
```

```
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
 <ListView
 android:id="@+id/ListView01"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
                                        ></ListView>

</LinearLayout>
```

编写文件 DeviceControl.java 来监听用户触摸选择列表的选项，根据用户选择的选项来到指定的界面。文件 DeviceControl.java 的具体实现代码如下所示。

```
public class DeviceControl extends Activity {
     public int n;
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.devicecontrol);

/*          Button buttonSearch = (Button)findViewById(R.id.SearchState);
         Button buttonUpdate = (Button)findViewById(R.id.UpdateState);
         Button buttonBack = (Button)findViewById(R.id.back);

         final TextView textView = (TextView)findViewById(R.id.StateOutput);
         n=(int)Math.random()*2;

         buttonSearch.setOnClickListener(new View.OnClickListener() {
              public void onClick(View view) {
                if (n==1)
                {
                     textView.setText("灯是亮着的");
                }

                if (n==0)
                {
                     textView.setText("灯是关着的");
                }
              }
           });

         buttonUpdate.setOnClickListener(new View.OnClickListener() {
              public void onClick(View view) {

                if (n==1)
                {
                     n=0;
                }

                if (n==0)
                {
                     n=1;
                }
```

```
            }
          });
        buttonBack.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
              finish();
            }
          });*/
        ListView listView=(ListView) findViewById(R.id.ListView01);
        List<String>list=new ArrayList<String>();
        list.add("温度");
        list.add("电灯");
        list.add("电扇");
        ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,list);

        listView.setAdapter(adapter);
        listView.setOnItemClickListener(new listener());



    }

    class listener implements OnItemClickListener{

        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                  long arg3) {
            // TODO Auto-generated method stub
            Intent intent;
            switch(arg2){
            case 0:
            intent= new Intent (DeviceControl.this,Temperature.class);
            startActivity(intent);
                  break;
            case 1:
                  intent = new Intent(DeviceControl.this,Lights.class);
                  startActivity(intent);
                  break;
            }
        }
    }
}
```

10.5.2　温度控制界面

当在电器控制主界面列表中选择“温度”选项后会来到温度控制界面，此界面的布局文件是temperature.xml，具体实现代码如下所示。

```
<AbsoluteLayout
android:id="@+id/widget0"
android:layout_width="fill_parent"
```

```
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<LinearLayout
android:id="@+id/widget36"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical"
android:layout_x="5px"
android:layout_y="3px"
>
</LinearLayout>
<Button
android:id="@+id/search"
android:layout_width="95px"
android:layout_height="wrap_content"
android:text="&#28201;&#24230;&#26597;&#35810;"
android:layout_x="11px"
android:layout_y="378px"
>
</Button>
<Button
android:id="@+id/update"
android:layout_width="102px"
android:layout_height="44px"
android:text="&#21047;&#26032;"
android:layout_x="113px"
android:layout_y="378px"
>
</Button>
<Button
android:id="@+id/back"
android:layout_width="99px"
android:layout_height="45px"
android:text="&#36820;&#22238;"
android:layout_x="221px"
android:layout_y="378px"
>
</Button>
<TextView
android:id="@+id/outcome"
android:layout_width="183px"
android:layout_height="44px"
android:text=""
android:layout_x="67px"
android:layout_y="160px"
>
</TextView>
</AbsoluteLayout>
```

编写 Temperature.java 文件，功能是监听用户的单击按钮在界面中显示当前温度，具体实现代码如下所示。

```
public class Temperature extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.temperature);

        Button buttonSearch = (Button)findViewById(R.id.search);
        Button buttonUpdate = (Button)findViewById(R.id.update);
        Button buttonBack = (Button)findViewById(R.id.back);

        final TextView textView = (TextView)findViewById(R.id.outcome);

        buttonSearch.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
            double n=Math.random()*100;
            textView.setText((int)n+"摄氏度");
            }
          });

        buttonUpdate.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
            double n=Math.random()*100;
            textView.setText((int)n+"摄氏度");
            }
          });
        buttonBack.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
              finish();
            }
          });
    }
```

温度控制界面的执行效果如图 10-4 所示。

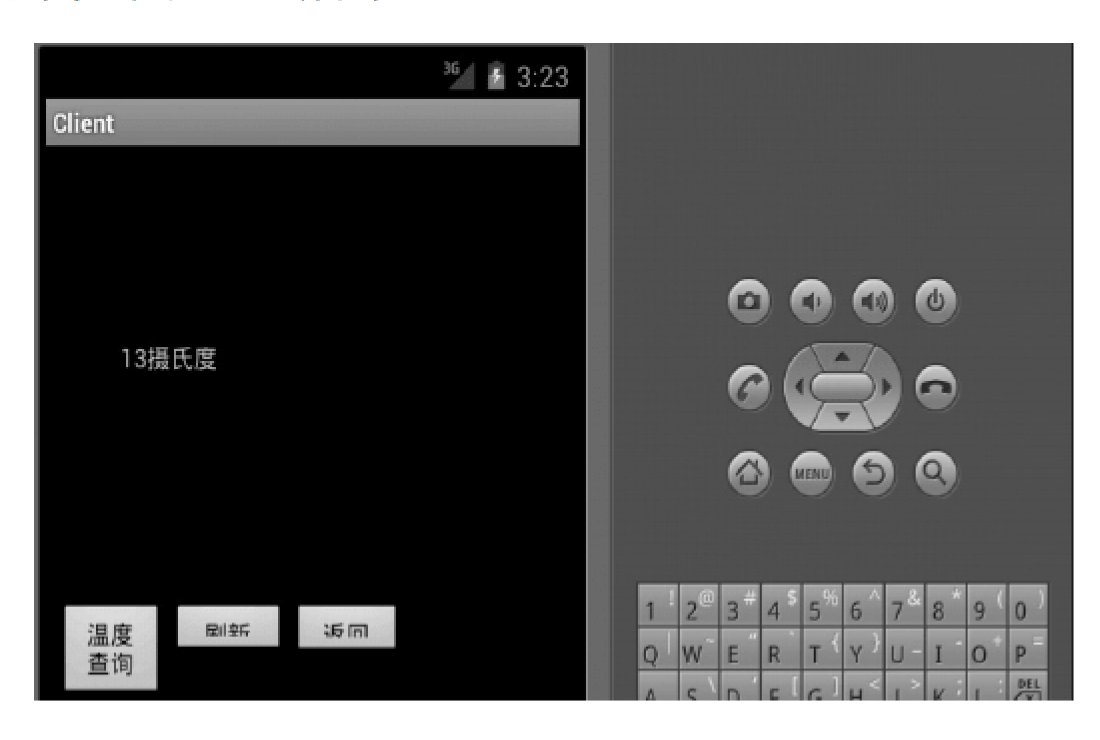
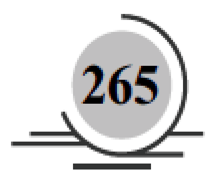


图 10-4　温度控制界面的执行效果

10.5.3　电灯控制界面

当在电器控制主界面列表中选择“电灯”选项后会来到电灯控制界面，此界面的布局文件是



light.xml，功能是在屏幕上方显示不同房间的电灯控制按钮，在下方显示操作按钮。文件 light.xml 的具体实现代码如下所示。

```
<AbsoluteLayout
android:id="@+id/widget0"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/keting_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#23458;&#21381;&#28783;"
android:layout_x="5px"
android:layout_y="5px"
>
</TextView>
<TextView
android:id="@+id/zhuwo_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#20027;&#21351;&#28783;"
android:layout_x="5px"
android:layout_y="76px"
>
</TextView>
<RadioGroup
android:id="@+id/radiogroup2"
android:layout_width="165px"
android:layout_height="42px"
android:orientation="horizontal"
android:gravity="center"
android:layout_x="10px"
android:layout_y="112px"
>
<RadioButton
android:id="@+id/radiobutton2_on"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#24320;"
android:textStyle="bold"
>
</RadioButton>
<RadioButton
android:id="@+id/radiobutton2_off"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#20851;"
android:textStyle="bold"
android:checked="true"
```

```
>
</RadioButton>
</RadioGroup>
<RadioGroup
android:id="@+id/radiogroup1"
android:layout_width="163px"
android:layout_height="44px"
android:orientation="horizontal"
android:gravity="center"
android:layout_x="10px"
android:layout_y="33px"
>
<RadioButton
android:id="@+id/radiobutton1_on"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#24320;"
android:textStyle="bold"
android:layout_gravity="center_vertical"
>
</RadioButton>
<RadioButton
android:id="@+id/radiobutton1_off"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="&#20851;"
android:textStyle="bold"
android:checked="true"
android:layout_gravity="center_vertical"
>
</RadioButton>
</RadioGroup>
<Button
android:id="@+id/getstatus"
android:layout_width="70px"
android:layout_height="wrap_content"
android:padding="5px"
android:text="&#33719;&#21462;&#29366;&#24577;"
android:textStyle="bold"
android:layout_x="5px"
android:layout_y="300px"
>
</Button>
<Button
android:id="@+id/refresh"
android:layout_width="70px"
android:layout_height="wrap_content"
android:text="&#21047;&#26032;&#26174;&#31034;"
android:textStyle="bold"
android:layout_x="78px"
android:layout_y="300px"
```

```
>
</Button>
<Button
android:id="@+id/reset"
android:layout_width="70px"
android:layout_height="wrap_content"
android:text="&#37325;&#26032;&#35774;&#23450;"
android:textStyle="bold"
android:layout_x="150px"
android:layout_y="300px"
>
</Button>
<Button
android:id="@+id/back"
android:layout_width="70px"
android:layout_height="wrap_content"
android:text="&#36820;&#22238;"
android:textStyle="bold"
android:layout_x="220px"
android:layout_y="300px"
>
</Button>
</AbsoluteLayout>
```

编写文件 Lights.java 获取用户对不同房间电灯的控制按钮值，并监听用户触摸单击操作按钮值，通过这两个值实现对家居电灯的控制管理功能。文件 Lights.java 的具体实现代码如下所示。

```
public class Lights extends Activity implements OnClickListener{

    private static RadioButton radioButton01;
    private static RadioButton radioButton02;
    @Override
    public void onClick(View arg0) {
        // TODO Auto-generated method stub
        switch(arg0.getId()) {
        case R.id.back:
        finish();
        Intent intent = new Intent(Lights.this,DeviceControl.class );
        startActivity(intent);

        break;
        case R.id.reset:

        SendSMS("13889609674",newStatus());
        Toast.makeText(this,newStatus(),Toast.LENGTH_LONG).show();
        break;
        }

    }
    private String newStatus(){
        StringBuilder   ns = new StringBuilder();
        ns.append("10*keting*");
```

```
            if(radioButton01.isChecked())
                    ns.append("1");
            if(radioButton02.isChecked())
                    ns.append("0");
            return ns.toString();


        }
        private void SendSMS(String telnumer,String content){
            Intent intent = new Intent(
                    "android.provider.Telephony.SMS_SEND");
            SmsManager smsManager = SmsManager.getDefault();
            PendingIntent   mpi = PendingIntent.getBroadcast(
                    this,0,intent,PendingIntent.FLAG_ONE_SHOT);
            smsManager.sendTextMessage(telnumer,null,
                    content,mpi,null);


        }
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            setContentView(R.layout.light);
            Button backButton=(Button)findViewById(R.id.back);
            Button refreshButton=(Button)findViewById(R.id.refresh);
            Button getButton=(Button)findViewById(R.id.getstatus);
            Button setButton=(Button)findViewById(R.id.reset);
            radioButton01=(RadioButton)findViewById(R.id.radiobutton1_on);
            radioButton02=(RadioButton)findViewById(R.id.radiobutton1_off);
            backButton.setOnClickListener(this);
            refreshButton.setOnClickListener(this);
            getButton.setOnClickListener(this);
            setButton.setOnClickListener(this);
        }
}
```

电灯控制界面的执行效果如图 10-5 所示。

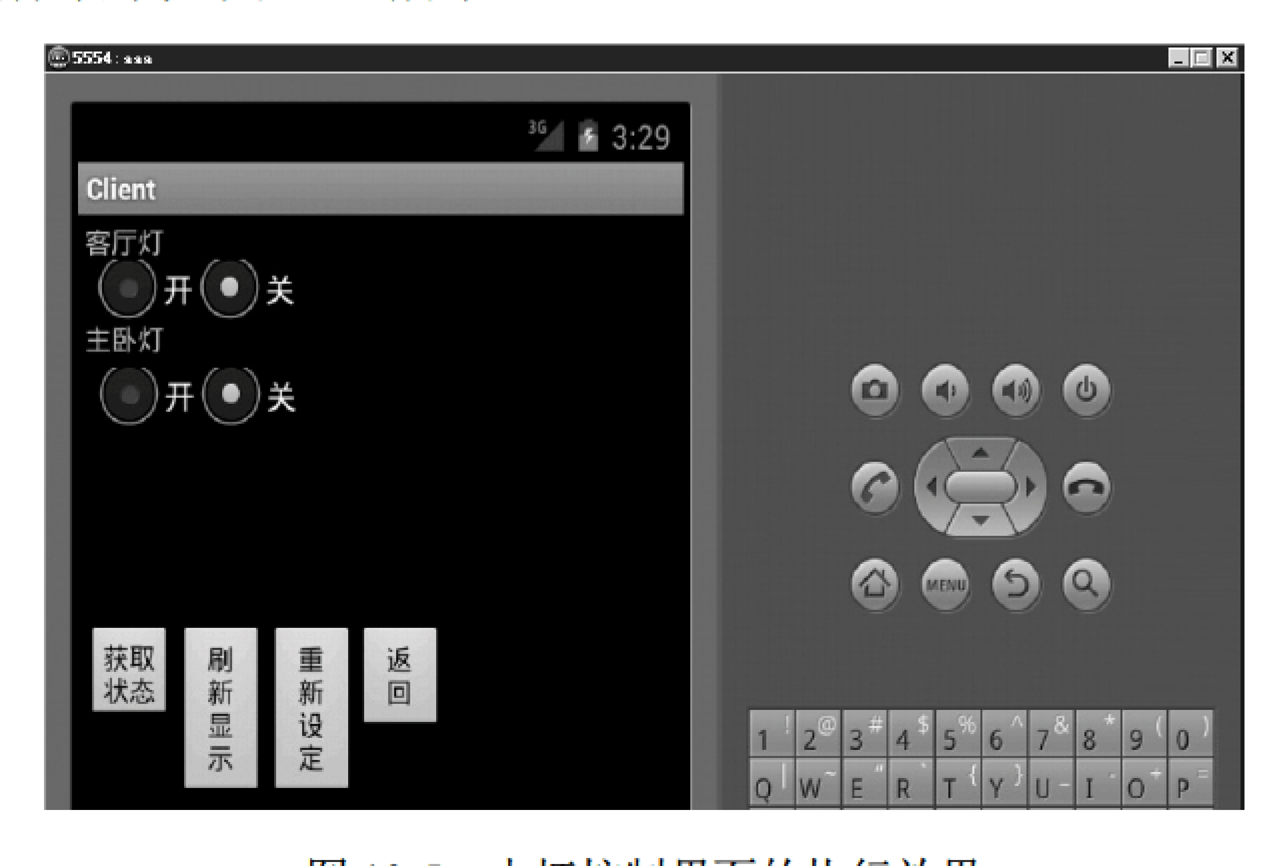


图 10-5 电灯控制界面的执行效果

10.6 预案管理模块

知识点讲解：光盘:视频\知识点\第 10 章\预案管理模块.avi

为了方便系统管理，在本智能家居系统中预先设置了预案管理模块。通过预案管理模块，可以快速地对天气情况、历史数据和系统设置信息进行浏览并管理。在本节的内容中，将详细讲解本系统预案管理模块的具体实现流程。

10.6.1 天气情况

编写 lx_weather.xml 文件在屏幕中构建一个天气预报界面，通过传感器获取当前的温度、湿度和 4 天的天气情况。lx_weather.xml 文件的具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
     <TextView    android:id="@+id/tab_weather_current_condition"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Temperature:60, Humidity:64%">
    </TextView>
     <TextView    android:id="@+id/tab_weather_current_wind"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Wind:N at 2 mph, 2009-09-20 12:21:00 +0000">
    </TextView>

    <ImageView android:id="@+id/tab_weather_current_image"
        android:src="@drawable/sunny"
        android:layout_gravity="center_vertical|center_horizontal"
        android:layout_marginBottom="8dip" android:layout_marginTop="15dip"

android:layout_height="100dip" android:layout_width="100dip"/>

    <TextView    android:id="@+id/tab_weather_current_city"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="New York, NY"
         android:gravity="center_vertical|center_horizontal">
    </TextView>

    <LinearLayout android:layout_width="fill_parent"
         android:layout_height="fill_parent"
         android:orientation="horizontal" android:layout_marginTop="20dip">
        <LinearLayout    android:orientation="vertical"
```

```
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_weight="1">
                <TextView   android:id="@+id/tab_weather_d1_date"
                      android:layout_width="fill_parent"
                      android:layout_height="wrap_content"
                      android:text="Mon"
                      android:gravity="center_horizontal">
           </TextView>
           <ImageView android:id="@+id/tab_weather_d1_image"
                           android:layout_marginTop="3dip"
                           android:layout_marginBottom="3dip"
                           android:src="@drawable/sunny"


android:layout_gravity="center_vertical|center_horizontal"
                           android:layout_height="40dip"
                           android:layout_width="40dip"/>
           <TextView   android:id="@+id/tab_weather_d1_temperature"
                      android:layout_width="fill_parent"
                      android:layout_height="wrap_content"
                      android:text="15/29"
                      android:gravity="center_horizontal">
           </TextView>
         </LinearLayout>
         <LinearLayout   android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_weight="1">
                <TextView   android:id="@+id/tab_weather_d2_date"
                      android:layout_width="fill_parent"
                      android:layout_height="wrap_content"
                      android:text="Tue"
                      android:gravity="center_horizontal">
           </TextView>
           <ImageView android:id="@+id/tab_weather_d2_image"
                           android:layout_marginTop="3dip"
                           android:layout_marginBottom="3dip"
                           android:src="@drawable/sunny"


android:layout_gravity="center_vertical|center_horizontal"
                           android:layout_height="40dip"
                           android:layout_width="40dip"/>
           <TextView   android:id="@+id/tab_weather_d2_temperature"
                      android:layout_width="fill_parent"
                      android:layout_height="wrap_content"
                      android:text="15/29"
                      android:gravity="center_horizontal">
           </TextView>
         </LinearLayout>
```

```
        <LinearLayout   android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_weight="1">
                <TextView   android:id="@+id/tab_weather_d3_date"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="Wed"
                    android:gravity="center_horizontal">
          </TextView>
          <ImageView android:id="@+id/tab_weather_d3_image"
                         android:layout_marginTop="3dip"
                         android:layout_marginBottom="3dip"
                         android:src="@drawable/sunny"


android:layout_gravity="center_vertical|center_horizontal"
                         android:layout_height="40dip"
                         android:layout_width="40dip"/>
          <TextView   android:id="@+id/tab_weather_d3_temperature"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="15/29"
                    android:gravity="center_horizontal">
          </TextView>
        </LinearLayout>
        <LinearLayout   android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_weight="1">
                 <TextView   android:id="@+id/tab_weather_d4_date"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="Thu"
                    android:gravity="center_horizontal">
          </TextView>
          <ImageView android:id="@+id/tab_weather_d4_image"
                         android:layout_marginTop="3dip"
                         android:layout_marginBottom="3dip"
                         android:src="@drawable/sunny"


android:layout_gravity="center_vertical|center_horizontal"
                         android:layout_height="40dip"
                         android:layout_width="40dip"/>
          <TextView   android:id="@+id/tab_weather_d4_temperature"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="15/29"
                    android:gravity="center_horizontal">
          </TextView>
```

```
            </LinearLayout>
        </LinearLayout>
</LinearLayout>
```

编写文件 Lx_weather.java 获取指定城市的天气信息，并监听用户触摸是否按下设备上的 MENU 按键。文件 Lx_weather.java 的具体实现代码如下所示。

```
public class Lx_weather extends Activity{
     final static int MENU_START_SERVICE= Menu.FIRST ;
     final static int MENU_STOP_SERVICE = Menu.FIRST + 1;
     final static int MENU_REFRESH = Menu.FIRST + 2;
     final static int MENU_QUIT = Menu.FIRST +3;

     private lx_DBAdapter dbAdapter ;

      @Override
          public void onCreate(Bundle savedInstanceState) {
               super.onCreate(savedInstanceState);
               setContentView(R.layout.lx_weather);

               dbAdapter = new lx_DBAdapter(this);
               dbAdapter.open();
                 dbAdapter.LoadConfig();

      }

       @Override
      public boolean onCreateOptionsMenu(Menu menu){

           menu.add(0,MENU_START_SERVICE,0,"启动服务");
           menu.add(0,MENU_STOP_SERVICE,1,"停止服务");
           menu.add(0,MENU_REFRESH ,2,"刷新");
           menu.add(0,MENU_QUIT,3,"退出");
           return true;
      }

      @Override
      public boolean onOptionsItemSelected(MenuItem item){
           final Intent serviceIntent = new Intent(this, lx_WeatherService.class);
           switch(item.getItemId()){
                case MENU_REFRESH:
                     RefreshWeatherData();
                return true;
                case MENU_START_SERVICE:
                     startService(serviceIntent);
                return true;
                case MENU_STOP_SERVICE:
                     stopService(serviceIntent);
                return true;
            case MENU_QUIT:
                finish();
                break;
```

```
        }
        return false;
    }

    private void RefreshWeatherData(){

        //当前温度
            TextView currentCondition = (TextView)findViewById(R.id.tab_weather_current_condition);
            TextView currentWind = (TextView)findViewById(R.id.tab_weather_current_wind);
            ImageView currentImage = (ImageView)findViewById(R.id.tab_weather_current_image);
            TextView currentCity = (TextView)findViewById(R.id.tab_weather_current_city);

            String msgCondition = "";
            msgCondition += "Temperature：" + lx_Weather.current_temp + ", ";
            msgCondition += lx_Weather.current_humidity ;
            currentCondition.setText(msgCondition);

            currentWind.setText(lx_Weather.current_wind + ", " + lx_Weather.current_date_time);
            currentImage.setImageBitmap(lx_Weather.current_image);
            currentCity.setText(lx_Weather.city);

             //预报：第 1 天
            TextView forcastD1Date = (TextView)findViewById(R.id.tab_weather_d1_date);
            ImageView forcastD1Image = (ImageView)findViewById(R.id.tab_weather_d1_image);
            TextView forcastD1Temperature = (TextView)findViewById(R.id.tab_weather_d1_temperature);

            forcastD1Date.setText(lx_Weather.day[0].day_of_week);
            forcastD1Image.setImageBitmap(lx_Weather.day[0].image);

            String msgD1Temperature = lx_Weather.day[0].high + "/" + lx_Weather.day[0].low;
            forcastD1Temperature.setText(msgD1Temperature);

             //预报：第 2 天
            TextView forcastD2Date = (TextView)findViewById(R.id.tab_weather_d2_date);
            ImageView forcastD2Image = (ImageView)findViewById(R.id.tab_weather_d2_image);
            TextView forcastD2Temperature = (TextView)findViewById(R.id.tab_weather_d2_temperature);

            forcastD2Date.setText(lx_Weather.day[1].day_of_week);
            forcastD2Image.setImageBitmap(lx_Weather.day[1].image);

            String msgD2Temperature = lx_Weather.day[1].high + "/" + lx_Weather.day[1].low;
            forcastD2Temperature.setText(msgD2Temperature);

             //预报：第 3 天
            TextView forcastD3Date = (TextView)findViewById(R.id.tab_weather_d3_date);
            ImageView forcastD3Image = (ImageView)findViewById(R.id.tab_weather_d3_image);
            TextView forcastD3Temperature = (TextView)findViewById(R.id.tab_weather_d3_temperature);

            forcastD3Date.setText(lx_Weather.day[2].day_of_week);
            forcastD3Image.setImageBitmap(lx_Weather.day[2].image);
```

```
            String msgD3Temperature = lx_Weather.day[2].high + "/" + lx_Weather.day[2].low;
            forcastD3Temperature.setText(msgD3Temperature);


             //预报：第 4 天
            TextView forcastD4Date = (TextView)findViewById(R.id.tab_weather_d4_date);
            ImageView forcastD4Image = (ImageView)findViewById(R.id.tab_weather_d4_image);
            TextView forcastD4Temperature = (TextView)findViewById(R.id.tab_weather_d4_temperature);

            forcastD4Date.setText(lx_Weather.day[3].day_of_week);
            forcastD4Image.setImageBitmap(lx_Weather.day[3].image);

            String msgD4Temperature = lx_Weather.day[3].high + "/" + lx_Weather.day[3].low;
            forcastD4Temperature.setText(msgD4Temperature);

    }

}
```

编写文件 Weather01.java 在屏幕顶端显示“天气预报”、“历史数据”和“系统设置”3 个选项卡，具体实现代码如下所示。

```
public class Weather01 extends TabActivity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);

        TabHost tabHost=getTabHost();
        tabHost.addTab(tabHost.newTabSpec("TAB1").setIndicator("天气预报",getResources().
            getDrawable(R.drawable.tab_weather)).setContent(new Intent(this,Lx_weather.class)));
        tabHost.addTab(tabHost.newTabSpec("TAB2").setIndicator("历史数据",getResources().
            getDrawable(R.drawable.tab_history)).setContent(new Intent(this,Lx_history.class)));
        tabHost.addTab(tabHost.newTabSpec("TAB3").setIndicator("系统设置",getResources().
            getDrawable(R.drawable.tab_setup)).setContent(new Intent(this,Lx_setup.class)));
    }
}
```

编写文件 lx_WeatherAdapter.java，功能是在线获取当前指定城市的天气信息，具体实现代码如下所示。

```
public class lx_WeatherAdapter {
     public static void GetWeatherData() throws IOException, Throwable {
     String queryString = "http://www.google.com/ig/api?weather=" + lx_Config.CityName;
     URL aURL = new URL(queryString.replace(" ", "%20"));
          URLConnection conn = aURL.openConnection();
        conn.connect();
        InputStream is = conn.getInputStream();

          XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
        factory.setNamespaceAware(true);
        XmlPullParser parser = factory.newPullParser();
```

```
        parser.setInput(is,"UTF-8");

        int dayCounter = 0;
        while(parser.next()!= XmlPullParser.END_DOCUMENT){
          String element = parser.getName();
          if (element != null && element.equals("forecast_information")){
                while(true){
                      int eventCode = parser.next();
                      element =    parser.getName();
                      if (eventCode == XmlPullParser.START_TAG){
                            if (element.equals("city")){
                            lx_Weather.city =    parser.getAttributeValue(0);
                            }else if (element.equals("current_date_time")){
                                  lx_Weather.current_date_time = parser.getAttributeValue(0);
                            }
                      }

                      if (element.equals("forecast_information") &&
                                  eventCode == XmlPullParser.END_TAG){
                            break;
                      }
                }
          }
          if (element != null && element.equals("current_conditions")){
                while(true){
                      int eventCode = parser.next();
                      element =    parser.getName();
                      if (eventCode == XmlPullParser.START_TAG){
                            if (element.equals("condition")){
                            lx_Weather.current_condition =    parser.getAttributeValue(0);
                            }else if (element.equals("temp_f")){
                                  lx_Weather.current_temp = parser.getAttributeValue(0);
                            }else if (element.equals("humidity")){
                                  lx_Weather.current_humidity = parser.getAttributeValue(0);
                            }else if (element.equals("wind_condition")){
                                  lx_Weather.current_wind = parser.getAttributeValue(0);
                            }else if (element.equals("icon")){
                                  lx_Weather.current_image_url = parser.getAttributeValue(0);
                                  lx_Weather.current_image = GetURLBitmap(lx_Weather.current_image_url);
                            }
                      }

                      if (element.equals("current_conditions") &&
                                  eventCode == XmlPullParser.END_TAG){
                            break;
                      }
                }
          }
          if (element != null && element.equals("forecast_conditions")){
                while(true){
```

```
                    int eventCode = parser.next();
                    element =   parser.getName();
                    if (eventCode == XmlPullParser.START_TAG){
                        if (element.equals("day_of_week")){
                        lx_Weather.day[dayCounter].day_of_week =   parser.getAttributeValue(0);
                        }else if (element.equals("low")){
                            lx_Weather.day[dayCounter].low = parser.getAttributeValue(0);
                        }else if (element.equals("high")){



          lx_Weather.day[dayCounter].high = parser.getAttributeValue(0);
                        }else if (element.equals("icon")){
                            lx_Weather.day[dayCounter].image_url = parser.getAttributeValue(0);
                            lx_Weather.day[dayCounter].image = GetURLBitmap(lx_Weather.day[dayCounter].
image_url);
                        }else if (element.equals("condition")){
                            lx_Weather.day[dayCounter].condition = parser.getAttributeValue(0);
                        }
                    }

                    if (element.equals("forecast_conditions") &&
                            eventCode == XmlPullParser.END_TAG){
                        dayCounter++;
                        break;
                    }
                }
            }
        }
        is.close();
    }
    private static Bitmap GetURLBitmap(String urlString){

        URL url = null;
        Bitmap bitmap = null;
        try {
         url = new URL("http://www.google.com" + urlString);
        }
        catch (MalformedURLException e){
           e.printStackTrace();
        }
        try{
           HttpURLConnection conn = (HttpURLConnection) url.openConnection();
           conn.connect();
           InputStream is = conn.getInputStream();
           bitmap = BitmapFactory.decodeStream(is);
           is.close();
        }
        catch (IOException e){
           e.printStackTrace();
        }
        return bitmap;
```

```
        }
}
```

编写文件 lx_WeatherService.java，功能是根据用户选择的选项来启动或停止当前的天气服务活动，具体实现代码如下所示。

```
public class lx_WeatherService extends Service{
    private lx_DBAdapter dbAdapter ;
    private Thread workThread;
    private static ArrayList<lx_SimpleSms> smsList = new ArrayList<lx_SimpleSms>();
    private static int timeCounter = 1;

    public static void RequerSMSService(lx_SimpleSms sms){
        if (lx_Config.ProvideSmsService.equals("true")){
            smsList.add(sms);
        }
    }
    private void SaveSmsData(lx_SimpleSms sms){
        if (lx_Config.SaveSmsInfo.equals("true")){
            dbAdapter.SaveOneSms(sms);
        }
    }
    @Override
    public void onCreate() {
        super.onCreate();

        dbAdapter = new lx_DBAdapter(this);
        dbAdapter.open();

        Toast.makeText(this, "启动天气服务", Toast.LENGTH_LONG).show();
        workThread = new Thread(null,backgroudWork,"WorkThread");
    }
    @Override
    public void onStart(Intent intent, int startId) {
        super.onStart(intent, startId);
        if (!workThread.isAlive()){
            workThread.start();
        }
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "天气服务启动停止", Toast.LENGTH_SHORT).show();
        workThread.interrupt();
    }
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    private Runnable backgroudWork = new Runnable(){
```

```
			@Override
			public void run() {
				try {
					while(!Thread.interrupted()){
						ProcessSmsList();
						GetGoogleWeatherData();

						Thread.sleep(1000);
					}
				} catch (InterruptedException e) {
					e.printStackTrace();
				}
			}
		};
		private void ProcessSmsList(){
			if (smsList.size()==0){
				return;
			}
			SmsManager smsManager = SmsManager.getDefault();
			PendingIntent mPi = PendingIntent.getBroadcast(this, 0, new Intent(), 0);
			while(smsList.size()>0){
				lx_SimpleSms sms = smsList.get(0);
				smsList.remove(0);
				smsManager.sendTextMessage(sms.Sender, null, lx_Weather.GetSmsMsg(), mPi, null);
				sms.ReturnResult = lx_Weather.GetSmsMsg();
				SaveSmsData(sms);

			}
		}
		private void GetGoogleWeatherData(){
			Log.i("TIMER",String.valueOf(timeCounter));
			if (timeCounter-- < 0){
				timeCounter = Integer.parseInt(lx_Config.RefreshSpeed);
				Log.i("TIMER","NOW");
				try {
					lx_WeatherAdapter.GetWeatherData();
				} catch (IOException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				} catch (Throwable e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}
		}
}
```

天气情况模块执行后面的效果如图 10-6 所示。

用户触摸按下设备 MENU 按键后的执行效果如图 10-7 所示。




图 10-6　天气情况模块的执行效果



图 10-7　按下 MENU 按键后的执行效果

10.6.2　历史数据

当在屏幕顶部选择“历史数据”选项后会来到系统的历史数据界面，如图 10-8 所示。



图 10-8　历史数据界面

编写文件 lx_history.xml，功能是通过 ListView 控件列表显示系统内的历史短信信息，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/black">

     <TextView    android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="SQLite 数据库中的短信服务信息：">
         </TextView>
     <ListView android:id="@android:id/list"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
         android:layout_marginTop="2dip">
```

```
    </ListView>
</LinearLayout>
```

编写文件 Lx_history.java 获取系统数据库中的历史信息，并且根据监听用户触摸按下设备的 MENU 按键值来进行对应的操作。文件 Lx_history.java 的具体实现代码如下所示。

```
public class Lx_history extends ListActivity{
    final static int MENU_REFRESH = Menu.FIRST;
    final static int MENU_DELETE = Menu.FIRST+1;
    final static int MENU_QUIT = Menu.FIRST+2;

    private lx_DBAdapter dbAdapter ;
    private lx_SmsAdapter dataAdapter;
     @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.lx_history);

            dbAdapter = new lx_DBAdapter(this);
             dbAdapter.open();

             dataAdapter = new lx_SmsAdapter(this);
            setListAdapter(dataAdapter);
     }

     @Override
     public boolean onCreateOptionsMenu(Menu menu){
         menu.add(0,MENU_REFRESH ,0,"刷新");
         menu.add(0,MENU_DELETE,1,"清空数据");
         menu.add(0,MENU_QUIT,1,"退出");
         return true;
     }

     @Override
     public boolean onOptionsItemSelected(MenuItem item){
         switch(item.getItemId()){
             case MENU_REFRESH:
                 lx_SmsAdapter.RefreshData();
                 setListAdapter(dataAdapter);
             return true;
         case MENU_DELETE:
             dbAdapter.DeleteAllSms();
             return true;
         case MENU_QUIT:
             finish();
             break;
         }
         return false;
      }
}
```

当触摸选择某一条历史数据时会调用文件 lx_SmsAdapter.java 显示这条数据的详细信息，具体实现代码如下所示。

```
public class lx_SmsAdapter extends BaseAdapter{
        private LayoutInflater mInflater;
        private static lx_DBAdapter dbAdapter ;
        private static lx_SimpleSms[ ] smsList ;

        public lx_SmsAdapter(Context context)
        {

          mInflater = LayoutInflater.from(context);
          dbAdapter = new lx_DBAdapter(context);
          dbAdapter.open();
          smsList = dbAdapter.GetAllSms();
        }

        public static void RefreshData(){
              smsList = dbAdapter.GetAllSms();
        }

      @Override
      public int getCount() {
            if (smsList == null)
                  return 0;
            else
                  return smsList.length;
      }

      @Override
      public Object getItem(int position) {
            if (smsList == null)
                  return 0;
            else
                  return smsList[position];
      }

      @Override
      public long getItemId(int position) {
            return position;
      }

      @Override
      public View getView(int position, View convertView, ViewGroup parent) {
            ViewHolder holder;

           if(convertView == null){
             convertView = mInflater.inflate(R.layout.lx_datarow, null);
             holder = new ViewHolder();
             holder.textRow01 = (TextView) convertView.findViewById(R.id.data_row_01);
             holder.textRow02 = (TextView) convertView.findViewById(R.id.data_row_02);

             convertView.setTag(holder);
           }
```

```
            else{
               holder = (ViewHolder) convertView.getTag();
            }


             if (smsList != null){
                    String row01Msg =" （"+position+"） " +" 发送者："+ smsList[position].Sender+"，"+smsList[position].
ReceiveTime;
                    holder.textRow01.setText(row01Msg);
                    holder.textRow02.setText(smsList[position].ReturnResult);
             }
            return convertView;
      }

      private class ViewHolder{
            TextView textRow01;
            TextView textRow02;
      }
}
```

10.6.3　系统设置

当在屏幕顶部选择“系统设置”选项后会来到系统设置界面，如图 10-9 所示。



图 10-9　系统设置界面的执行效果

编写文件 lx_setup.xml，功能是通过文本框控件和复选框显示系统设置信息，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      >
      <LinearLayout android:orientation="horizontal"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content">
              <TextView    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="城市名称（拼音）：">
              </TextView>
             <EditText android:id="@+id/tab_setup_city_name"
```

```
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="Haerbin" >
            </EditText>
        </LinearLayout>

         <LinearLayout android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="更新频率：">
            </TextView>
            <EditText android:id="@+id/tab_setup_refresh_speed"
                android:layout_width="120dip"
                android:layout_height="wrap_content"
                android:numeric="integer"
                android:text="600" >
            </EditText>
            <TextView   android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="秒/次">
            </TextView>
        </LinearLayout>

        <LinearLayout android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
            <TextView   android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="提供短信服务：">
            </TextView>

            <CheckBox android:id="@+id/tab_setup_sms_service"
                android:text="是"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:checked="true">
            </CheckBox>
        </LinearLayout>

        <LinearLayout android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
            <TextView   android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="记录短信服务数据信息：">
            </TextView>

            <CheckBox android:id="@+id/tab_setup_save_sms_info"
```

```
                android:text="是"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:checked="true">
            </CheckBox>
        </LinearLayout>

         <LinearLayout android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="短信服务关键字：">
            </TextView>
            <EditText android:id="@+id/tab_setup_key_work"
                android:layout_width="120dip"
                android:layout_height="wrap_content"
                android:text="WR" >
            </EditText>
        </LinearLayout>

         <LinearLayout android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_marginTop="5dip">

            <Button android:id="@+id/tab_setup_apply"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="应用系统设置"
                android:layout_weight="1" android:layout_gravity="bottom">
            </Button>
            <Button android:id="@+id/tab_setup_cancel"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="取消系统设置"
                android:layout_weight="1" android:layout_gravity="bottom">
            </Button>
        </LinearLayout>
</LinearLayout>
```

编写文件 lx_Config.java，功能是预先设置系统设置变量的初始值，具体实现代码如下所示。

```
public class lx_Config {
    public static String CityName;
    public static String RefreshSpeed;
    public static String ProvideSmsService;
    public static String SaveSmsInfo;
    public static String KeyWord;
    public static void LoadDefaultConfig(){
        CityName = "济南";
```

```
            RefreshSpeed = "60";
            ProvideSmsService = "true";
            SaveSmsInfo = "true";
            KeyWord = "无敌";
      }
}
```

编写文件 lx_DBAdapter.java，功能是根据用户的设置值对数据库进行更新，具体实现代码如下所示。

```
        public void SaveConfig(){
              ContentValues updateValues = new ContentValues();
              updateValues.put(KEY_CITY_NAME, lx_Config.CityName);
              updateValues.put(KEY_REFRESH_SPEED, lx_Config.RefreshSpeed);
              updateValues.put(KEY_SMS_SERVICE, lx_Config.ProvideSmsService);
              updateValues.put(KEY_SMS_INFO, lx_Config.SaveSmsInfo);
              updateValues.put(KEY_KEY_WORD, lx_Config.KeyWord);

              db.update(DB_TABLE_CONFIG, updateValues,   KEY_ID + "=" + DB_CONFIG_ID, null);

              Toast.makeText(context, "系统设置保存成功", Toast.LENGTH_SHORT).show();

        }

        public void LoadConfig() {
        Cursor result =   db.query(DB_TABLE_CONFIG, new String[ ] { KEY_ID, KEY_CITY_NAME, KEY_
REFRESH_ SPEED,
                          KEY_SMS_SERVICE, KEY_SMS_INFO, KEY_KEY_WORD},
                         KEY_ID + "=" + DB_CONFIG_ID, null, null, null, null);
        if (result.getCount() == 0 || !result.moveToFirst()){
                  return;
        }
        lx_Config.CityName = result.getString(result.getColumnIndex(KEY_CITY_NAME));
        lx_Config.RefreshSpeed =   result.getString(result.getColumnIndex(KEY_REFRESH_SPEED));
        lx_Config.ProvideSmsService = result.getString(result.getColumnIndex(KEY_SMS_SERVICE));
        lx_Config.SaveSmsInfo =   result.getString(result.getColumnIndex(KEY_SMS_INFO));
        lx_Config.KeyWord = result.getString(result.getColumnIndex(KEY_KEY_WORD));

        Toast.makeText(context, "系统设置读取成功", Toast.LENGTH_SHORT).show();
  }


        /** 静态 Helper 类，用于建立、更新和打开数据库*/
        private static class DBOpenHelper extends SQLiteOpenHelper {

              public DBOpenHelper(Context context, String name, CursorFactory factory, int version) {
                super(context, name, factory, version);
              }

              private static final String DB_CREATE_CONFIG = "create table " +
                DB_TABLE_CONFIG + " (" + KEY_ID + " integer primary key autoincrement, " +
                KEY_CITY_NAME+ " text not null, " + KEY_REFRESH_SPEED+ " text," +
                KEY_SMS_SERVICE +" text, " + KEY_SMS_INFO +   " text, " +
                KEY_KEY_WORD + " text);";
```

```
        private static final String DB_CREATE_SMS = "create table " +
          DB_TABLE_SMS + " (" + KEY_ID + " integer primary key autoincrement, " +
          KEY_SENDER+ " text not null, " + KEY_BODY+ " text, " +
          KEY_RECEIVE_TIME +" text, " +     KEY_RETURN_RESULT + " text);";

        @Override
        public void onCreate(SQLiteDatabase _db) {
          _db.execSQL(DB_CREATE_CONFIG);
          _db.execSQL(DB_CREATE_SMS);

          //初始化系统配置的数据表
          lx_Config.LoadDefaultConfig();
          ContentValues newValues = new ContentValues();
          newValues.put(KEY_CITY_NAME, lx_Config.CityName);
          newValues.put(KEY_REFRESH_SPEED, lx_Config.RefreshSpeed);
          newValues.put(KEY_SMS_SERVICE, lx_Config.ProvideSmsService);
          newValues.put(KEY_SMS_INFO, lx_Config.SaveSmsInfo);
          newValues.put(KEY_KEY_WORD, lx_Config.KeyWord);
          _db.insert(DB_TABLE_CONFIG, null, newValues);

        }

        @Override
        public void onUpgrade(SQLiteDatabase _db, int _oldVersion, int _newVersion) {
          _db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE_CONFIG);
          _db.execSQL("DROP TABLE IF EXISTS " + DB_CREATE_SMS);
          onCreate(_db);
        }
      }
```

系统设置界面的执行效果如图 10-10 所示。



图 10-10　系统设置界面的执行效果

到此为止，本章智能家居系统全部介绍完毕。有关本实例的详细源码，请读者参阅本书附带光盘中的源码。

第 11 章　健康专家——智能心率计

近几年来，各大科技巨头都纷纷推出了智能可穿戴设备。随着人们对新技术接受度的提高，可穿戴设备必将成为未来人们生活的必需品之一。开发可穿戴设备需要掌握硬件和软件技术，因为本书讲解的是 Android 可穿戴技术开发，所以本书内容将以应用软件程序开发为主。在本章的内容中，将详细讲解开发 Android 智能心率计的基本知识，为步入本书后面知识的学习打下基础。

11.1　什么是心率

知识点讲解：光盘:视频\知识点\第 11 章\什么是心率.avi

心率（Heart Rate）是用来描述心动周期的专业术语，是指心脏每分钟跳动的次数，以第一声音为准。心率，现代汉语将心率解释为“心脏跳动的频率”。频率就是在单位时间内，某件事情发生的次数。两种解释合起来就是，心脏在一定时间内跳动的次数，也就是在一定时间内，心脏跳动快慢的意思。

据科学研究发现，正常成年人安静时的心率有显著的个体差异，平均在 75 次/分左右（60～100 次/分之间）。心率可因年龄、性别及其他生理情况而不同。初生儿的心率很快，可达 130 次/分以上。在成年人中，女性的心率一般比男性稍快。同一个人，在安静或睡眠时心率减慢，运动时或情绪激动时心率加快，在某些药物或神经体液因素的影响下，会使心率发生加快或减慢。经常进行体力劳动和体育锻炼的人，平时心率较慢。近年，国内大样本健康人群调查发现：国人男性静息心率的正常范围为 50～95 次/分，女性为 55～95 次/分。所以，心率随年龄、性别和健康状况变化而变化。

健康成人的心率为 60～100 次/分，大多数为 60～80 次/分，女性稍快；3 岁以下的小孩常在 100 次/分以上；老年人偏慢。成人每分钟心率超过 100 次（一般不超过 160 次/分）或婴幼儿超过 150 次/分者，称为窦性心动过速，常见于正常人运动、兴奋、激动、吸烟、饮酒和喝浓茶后；也可见于发热、休克、贫血、甲亢、心力衰竭及应用阿托品、肾上腺素、麻黄素等。如果心率在 160～220 次/分，常称为阵发性心动过速。心率低于 60 次/分者（一般在 40 次/分以上），称为窦性心动过缓，可见于长期从事重体力劳动和运动员；病理性的见于甲状腺机能低下、颅内压增高、阻塞性黄疸，以及洋地黄、奎尼丁或心得安类药物过量或中毒。如心率低于 40 次/分，应考虑有房室传导阻滞。心率过快超过 160 次/分，或低于 40 次/分，大多见于心脏病病人，病人常有心悸、胸闷、心前区不适，应及早进行详细检查，以便针对病因进行治疗。心脏每次收缩时由心室向动脉输出的血量叫做每搏输出量，心脏每分钟输出的血量叫做每分输出量，正常人在安静状态下每搏输出量为 70 毫升，如果心率按每分钟 75 次计算的话，每分输出量约为 5250 毫升。心输出量的多少，是衡量心脏工作能力的一项指标。

11.2　开发一个 Android 版心率计

知识点讲解：光盘:视频\知识点\第 11 章\开发一个 Android 版心率计.avi

实例	功能	源码路径
实例 11-1	心率测试系统	光盘:\daima\11\heart

本实例的功能是，在 Android 系统中开发一个心率测试应用程序。本实例是通过低功耗蓝牙（BLE）技术来检测传感器变化的，根据检测到的变化数据来测试心率的变化。本应用程序实例会搜索附近的 BLE 设备，并连接到远程心脏速率传感器的服务，会监听信号（平均脉冲）和心脏心率的变化（心跳或 RR 数据），并且能够检测到呼吸率如何影响心脏心率变异的，并根据检测结果生成三维图形演示。在本实例中，使用 BLE 技术传输心脏速率的数据。因为具有开放的协议，所以可以较快地建立一个开发环境。因为 BLE 是从 Android 4.3 开始推出的，所以本项目需要运行在 Android 4.3 的设备上。

11.2.1　扫描蓝牙设备

本系统的主界面 Activity 是 DeviceScanActivity，对应界面布局文件是 actionbar_indeterminate_progress.xml，具体实现代码如下所示。

```
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_height="wrap_content"
        android:layout_width="56dp"
        android:minWidth="56dp">
    <ProgressBar
            android:layout_width="32dp"
            android:layout_height="32dp"
            android:layout_gravity="center"/>
</FrameLayout>
```

通过上述代码，在屏幕中显示进度控件来显示当前的扫描进度。

主界面 Activity 的实现文件是 DeviceScanActivity.java，功能是调用 UI 布局文件 actionbar_indeterminate_progress.xml 加载显示的控件，如果没有扫描到 BLE 蓝牙设备则输出 error_bluetooth_not_supported 提示，如果搜索到则列表显示蓝牙 BLE 设备。文件 DeviceScanActivity.java 的具体实现代码如下所示。

```
/**
 *扫描，并显示可用的蓝牙 BLE 设备
 */
public class DeviceScanActivity extends ListActivity {

    private static final int REQUEST_ENABLE_BT = 1;
    private static final long SCAN_PERIOD = 500;

    private BleDevicesAdapter leDeviceListAdapter;
    private BluetoothAdapter bluetoothAdapter;
```

```
    private Scanner scanner;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getActionBar().setTitle(R.string.title_devices);

        //使用此检查，以确定 BLE 是否支持在设备上。然后可以选择性地禁用 BLE 相关的功能
        if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
            Toast.makeText(this, R.string.ble_not_supported, Toast.LENGTH_SHORT).show();
            finish();
            return;
        }

        //初始化一个蓝牙适配器。对于 API 级别 18 以上，通过 BluetoothManager 得到一个 BluetoothAdapter
        final BluetoothManager bluetoothManager =
                (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
        bluetoothAdapter = bluetoothManager.getAdapter();

        // Checks if Bluetooth is supported on the device
        if (bluetoothAdapter == null) {
            Toast.makeText(this, R.string.error_bluetooth_not_supported, Toast.LENGTH_SHORT).show();
            finish();
            return;
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.gatt_scan, menu);
        if (scanner == null || !scanner.isScanning()) {
            menu.findItem(R.id.menu_stop).setVisible(false);
            menu.findItem(R.id.menu_scan).setVisible(true);
            menu.findItem(R.id.menu_refresh).setActionView(null);
        } else {
            menu.findItem(R.id.menu_stop).setVisible(true);
            menu.findItem(R.id.menu_scan).setVisible(false);
            menu.findItem(R.id.menu_refresh).setActionView(
                    R.layout.actionbar_indeterminate_progress);
        }
        return true;
    }
//根据监听到的用户操作执行对应的事件处理程序
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_scan:
                leDeviceListAdapter.clear();
                if (scanner == null) {
                    scanner = new Scanner(bluetoothAdapter, mLeScanCallback);
                    scanner.startScanning();
```

```
                invalidateOptionsMenu();
            }
            break;
        case R.id.menu_stop:
            if (scanner != null) {
                scanner.stopScanning();
                scanner = null;

                invalidateOptionsMenu();
            }
            break;
    }
    return true;
}

@Override
protected void onResume() {
    super.onResume();

    // Ensures Bluetooth is enabled on the device.  If Bluetooth is not currently enabled,
    // fire an intent to display a dialog asking the user to grant permission to enable it
    if (!bluetoothAdapter.isEnabled()) {
        final Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
        return;
    }

    init();
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // User chose not to enable Bluetooth
    if (requestCode == REQUEST_ENABLE_BT) {
        if (resultCode == Activity.RESULT_CANCELED) {
            finish();
        } else {
            init();
        }
    }
    super.onActivityResult(requestCode, resultCode, data);
}
//暂停扫描
@Override
protected void onPause() {
    super.onPause();

    if (scanner != null) {
        scanner.stopScanning();
        scanner = null;
```

```
        }
    }
//监听用户单击操作命令选项事件
    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        final BluetoothDevice device = leDeviceListAdapter.getDevice(position);
        if (device == null)
            return;

        final Intent intent = new Intent(this, DeviceServicesActivity.class);
        intent.putExtra(DeviceServicesActivity.EXTRAS_DEVICE_NAME, device.getName());
        intent.putExtra(DeviceServicesActivity.EXTRAS_DEVICE_ADDRESS, device.getAddress());
        startActivity(intent);
    }

    private void init() {
        if (leDeviceListAdapter == null) {
            leDeviceListAdapter = new BleDevicesAdapter(getBaseContext());
            setListAdapter(leDeviceListAdapter);
        }

        if (scanner == null) {
            scanner = new Scanner(bluetoothAdapter, mLeScanCallback);
            scanner.startScanning();
        }

        invalidateOptionsMenu();
    }

    // 扫描设备回调
    private BluetoothAdapter.LeScanCallback mLeScanCallback =
            new BluetoothAdapter.LeScanCallback() {

                @Override
                public void onLeScan(final BluetoothDevice device, final int rssi, byte[ ] scanRecord) {
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            leDeviceListAdapter.addDevice(device, rssi);
                            leDeviceListAdapter.notifyDataSetChanged();
                        }
                    });
                }
            };

    private static class Scanner extends Thread {
        private final BluetoothAdapter bluetoothAdapter;
        private final BluetoothAdapter.LeScanCallback mLeScanCallback;

        private volatile boolean isScanning = false;
```

```
        Scanner(BluetoothAdapter adapter, BluetoothAdapter.LeScanCallback callback) {
            bluetoothAdapter = adapter;
            mLeScanCallback = callback;
        }

        public boolean isScanning() {
            return isScanning;
        }

        public void startScanning() {
            synchronized (this) {
                isScanning = true;
                start();
            }
        }

        public void stopScanning() {
            synchronized (this) {
                isScanning = false;
                bluetoothAdapter.stopLeScan(mLeScanCallback);
            }
        }

        @Override
        public void run() {
            try {
                while (true) {
                    synchronized (this) {
                        if (!isScanning)
                            break;

                        bluetoothAdapter.startLeScan(mLeScanCallback);
                    }

                    sleep(SCAN_PERIOD);

                    synchronized (this) {
                        bluetoothAdapter.stopLeScan(mLeScanCallback);
                    }
                }
            } catch (InterruptedException ignore) {
            } finally {
                bluetoothAdapter.stopLeScan(mLeScanCallback);
            }
        }
    }
}
```

通过上述代码可知，在扫描过程中可以控制扫描操作，方法是单击设备右上角的 stop、scan 按钮。执行效果如图 11-1 所示。

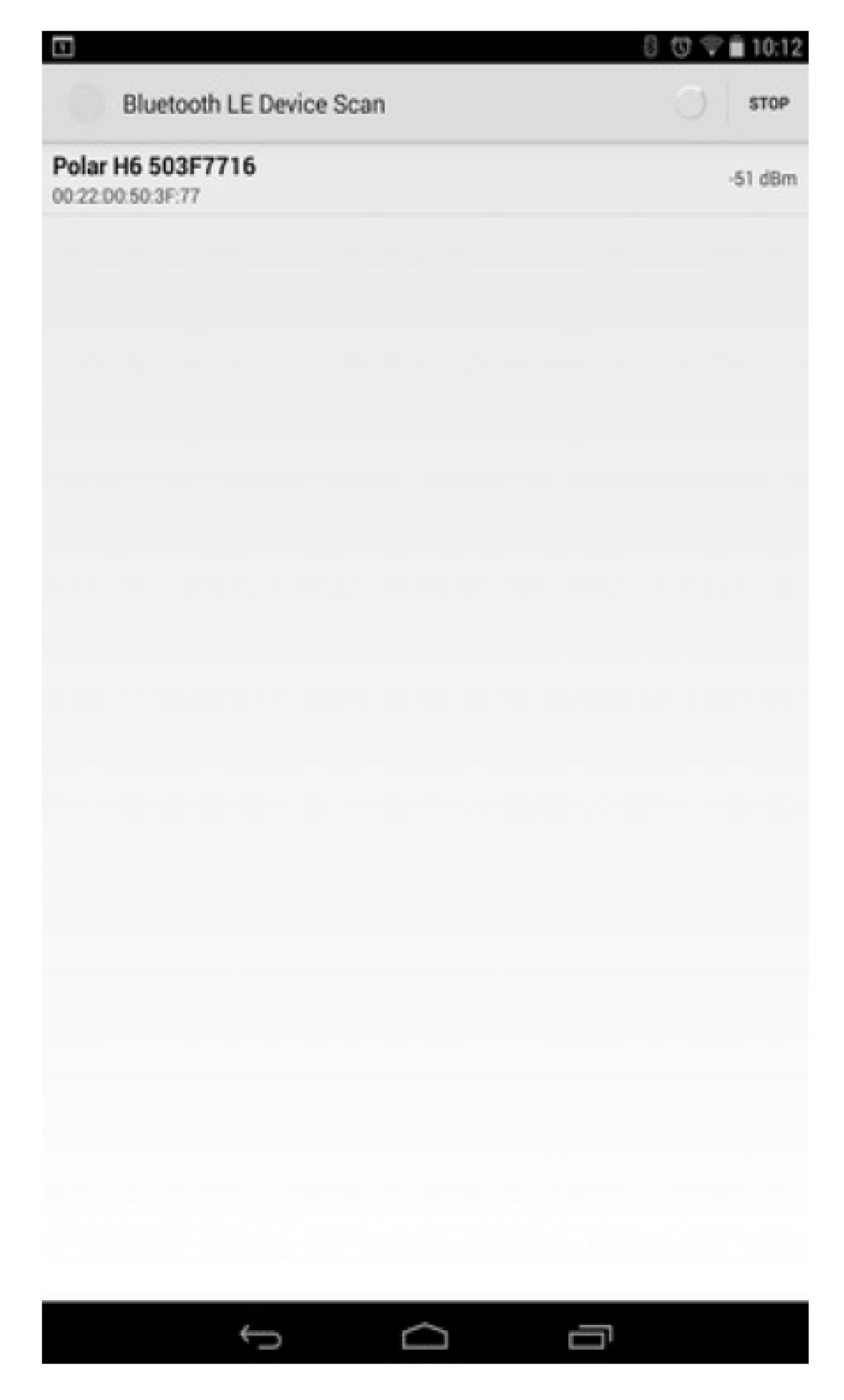


图 11-1　扫描到的蓝牙 BLE 设备列表

11.2.2　蓝牙控制界面

在主界面中列表显示扫描到的蓝牙 BLE 设备列表，单击列表中的某一个蓝牙设备后会来到蓝牙控制界面，在此界面可以设置和这个蓝牙设备的连接及断开连接操作，并显示蓝牙设备的 GAP 和 GATT 信息。蓝牙控制界面的 UI 布局文件是 gatt_services_characteristics.xml，具体实现代码如下所示。

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingTop="10dp">
    <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/label_device_address"
                android:textAppearance="?android:textAppearanceMedium"
                android:layout_marginRight="5dp"/>
        <TextView
```

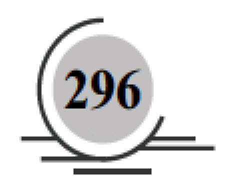
```
                android:id="@+id/device_address"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAppearance="?android:textAppearanceMedium"/>
    </LinearLayout>
   <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:textAppearanceMedium"
                android:text="@string/label_state"
                android:layout_marginRight="5dp"/>
        <TextView
                android:id="@+id/connection_state"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAppearance="?android:textAppearanceMedium"/>
    </LinearLayout>
    <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:textAppearanceMedium"
                android:text="@string/label_data"
                android:layout_marginRight="5dp"/>
        <TextView
                android:id="@+id/data_value"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAppearance="?android:textAppearanceMedium"
                android:text="@string/no_data"
                android:minLines="3"/>
     </LinearLayout>
     <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <TextView
                android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"
                android:textAppearance="?android:textAppearanceMedium"
                android:text="@string/label_heartrate"
                android:layout_marginRight="5dp"/>
        <TextView
                android:id="@+id/heartrate_value"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAppearance="?android:textAppearanceMedium"
                android:text="@string/no_data"
                android:minLines="1"/>
    </LinearLayout>
    <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <Button
            android:id="@+id/demo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/action_demo"
            android:clickable="false"
            android:focusable="false" />
    </LinearLayout>
    <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/label_services"
            android:paddingTop="4dp"
            android:paddingBottom="4dp"
            android:paddingLeft="10dp"
            android:paddingRight="10dp"
            android:textStyle="bold"
            android:textAppearance="?android:textAppearanceMedium"
            android:background="@android:drawable/divider_horizontal_bright" />
    <ExpandableListView
            android:id="@+id/gatt_services_list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:groupIndicator="@null"/>
</LinearLayout>
```

蓝牙控制界面的程序文件是 DeviceServicesActivity.java，功能是调用 UI 文件中的布局控件，并根据用户所选的蓝牙 BLE 设备提供连接和显示数据界面，并显示设备支持的 GATT 服务和特性，并且通过此 BleService 服务，可以实现与蓝牙 BLE 的 API 实现交互通信功能。

文件 DeviceServicesActivity.java 的具体实现代码如下所示。

```
public class DeviceServicesActivity extends Activity {
    private final static String TAG = DeviceServicesActivity.class.getSimpleName();
```



```
public static final String EXTRAS_DEVICE_NAME = "DEVICE_NAME";
public static final String EXTRAS_DEVICE_ADDRESS = "DEVICE_ADDRESS";

private TextView connectionState;
private TextView dataField;
private TextView heartRateField;
private TextView intervalField;
private Button demoButton;

private ExpandableListView gattServicesList;
private BleServicesAdapter gattServiceAdapter;

private String deviceName;
private String deviceAddress;
private BleService bleService;
private boolean isConnected = false;

private BleSensor<?> activeSensor;
private BleSensor<?> heartRateSensor;

 private OnServiceItemClickListener serviceListener;

//代码管理服务的生命周期
private final ServiceConnection serviceConnection = new ServiceConnection() {

    @Override
    public void onServiceConnected(ComponentName componentName, IBinder service) {
        bleService = ((BleService.LocalBinder) service).getService();
        if (!bleService.initialize()) {
            Log.e(TAG, "Unable to initialize Bluetooth");
            finish();
        }
        // Automatically connects to the device upon successful start-up initialization.
        bleService.connect(deviceAddress);
    }

    @Override
    public void onServiceDisconnected(ComponentName componentName) {
        bleService = null;
    }
};

//处理蓝牙发射服务中的常见活动
// ACTION_GATT_CONNECTED: 连接一个 GATT 服务
// ACTION_GATT_DISCONNECTED: 断开一个 GATT 服务连接
// ACTION_GATT_SERVICES_DISCOVERED: 发现一个 GATT 服务
// ACTION_DATA_AVAILABLE: received data from the device.   This can be a result of read
//                              or notification operations
```

```
    private final BroadcastReceiver gattUpdateReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();
            if (BleService.ACTION_GATT_CONNECTED.equals(action)) {
                isConnected = true;
                updateConnectionState(R.string.connected);
                invalidateOptionsMenu();
            } else if (BleService.ACTION_GATT_DISCONNECTED.equals(action)) {
                isConnected = false;
                updateConnectionState(R.string.disconnected);
                invalidateOptionsMenu();
                clearUI();
            } else if (BleService.ACTION_GATT_SERVICES_DISCOVERED.equals(action)) {
                // Show all the supported services and characteristics on the user interface.
                displayGattServices(bleService.getSupportedGattServices());
                   enableHeartRateSensor();
            } else if (BleService.ACTION_DATA_AVAILABLE.equals(action)) {
                   displayData(intent.getStringExtra(BleService.EXTRA_SERVICE_UUID),
intent.getStringExtra(BleService.EXTRA_TEXT));

            }
        }
    };

    //如果给定了一个 GATT 服务选项，请检查它对应的支持功能，具体请参考 http://d.android.com/reference/
android/bluetooth/BluetoothGatt.html 列出的支持的特性功能列表
    private final ExpandableListView.OnChildClickListener servicesListClickListner =
            new ExpandableListView.OnChildClickListener() {
                @Override
                public boolean onChildClick(ExpandableListView parent, View v, int groupPosition,
                                            int childPosition, long id) {
                    if (gattServiceAdapter == null)
                        return false;
                    final BluetoothGattCharacteristic characteristic = gattServiceAdapter.getChild(groupPosition,
                    childPosition);
                    final BleSensor<?> sensor = BleSensors.getSensor(characteristic.getService().getUuid().
                    toString());
                    if (activeSensor != null)
                        bleService.enableSensor(activeSensor, false);
                    if (sensor == null) {
                        bleService.readCharacteristic(characteristic);
                        return true;
                    }
                    if (sensor == activeSensor)
                        return true;
                    activeSensor = sensor;
                    bleService.enableSensor(sensor, true);
                    return true;
```

```
                }
            };
    private final BleServicesAdapter.OnServiceItemClickListener demoClickListener = new BleServicesAdapter.
    OnServiceItem ClickListener() {
        @Override
        public void onDemoClick(BluetoothGattService service) {
          Log.d(TAG, "onDemoClick: service" +service.getUuid().toString());
            final BleSensor<?> sensor = BleSensors.getSensor(service.getUuid().toString());
            if (sensor == null)
                return;
            final Class<? extends DemoSensorActivity> demoClass;
            if (sensor instanceof BleHeartRateSensor)
                demoClass = DemoHeartRateSensorActivity.class;
            else
                return;
            final Intent demoIntent = new Intent();
            demoIntent.setClass(DeviceServicesActivity.this, demoClass);
            demoIntent.putExtra(DemoSensorActivity.EXTRAS_DEVICE_ADDRESS, deviceAddress);
            demoIntent.putExtra(DemoSensorActivity.EXTRAS_SENSOR_UUID, service.getUuid().toString());
            startActivity(demoIntent);
        }
        @Override
        public void onServiceEnabled(BluetoothGattService service, boolean enabled) {
            if (gattServiceAdapter == null)
                return;
            final BleSensor<?> sensor = BleSensors.getSensor(service.getUuid().toString());
            if (sensor == null)
                return;
            if (sensor == activeSensor)
                return;
            if (activeSensor != null)
                bleService.enableSensor(activeSensor, false);
            activeSensor = sensor;
            bleService.enableSensor(sensor, true);
        }
        @Override
        public void onServiceUpdated(BluetoothGattService service) {
            final BleSensor<?> sensor = BleSensors.getSensor(service.getUuid().toString());
            if (sensor == null)
                return;
            bleService.updateSensor(sensor);
        }
    };
    private void clearUI() {
        gattServicesList.setAdapter((SimpleExpandableListAdapter) null);
        dataField.setText(R.string.no_data);
          heartRateField.setText(R.string.no_data);
          intervalField.setText(R.string.no_data);
    }
```

```
 public void setServiceListener(OnServiceItemClickListener listener) {
         this.serviceListener = listener;
 }
@Override
public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.gatt_services_characteristics);
       final Intent intent = getIntent();
       deviceName = intent.getStringExtra(EXTRAS_DEVICE_NAME);
       deviceAddress = intent.getStringExtra(EXTRAS_DEVICE_ADDRESS);
       // 设置用户界面
       ((TextView) findViewById(R.id.device_address)).setText(deviceAddress);
       gattServicesList = (ExpandableListView) findViewById(R.id.gatt_services_list);
       gattServicesList.setOnChildClickListener(servicesListClickListner);
       connectionState = (TextView) findViewById(R.id.connection_state);
       dataField = (TextView) findViewById(R.id.data_value);
         heartRateField = (TextView) findViewById(R.id.heartrate_value);
         demoButton = (Button) findViewById(R.id.demo);
         demoButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                       Log.d(TAG, "onClick serviceListener: "+serviceListener);
                       if (serviceListener == null)
                              return;
                       final BluetoothGattService service = gattServiceAdapter.getHeartRateService();
                       serviceListener.onDemoClick(service);
                       Log.d(TAG, "set service listener");
                }
         });
         demoButton.setVisibility(View.VISIBLE);

       getActionBar().setTitle(deviceName);
       getActionBar().setDisplayHomeAsUpEnabled(true);
       final Intent gattServiceIntent = new Intent(this, BleService.class);
       bindService(gattServiceIntent, serviceConnection, BIND_AUTO_CREATE);
}
@Override
protected void onResume() {
       super.onResume();
       registerReceiver(gattUpdateReceiver, makeGattUpdateIntentFilter());
       if (bleService != null) {
             final boolean result = bleService.connect(deviceAddress);
             Log.d(TAG, "Connect request result=" + result);
       }
}
@Override
protected void onPause() {
       super.onPause();
       unregisterReceiver(gattUpdateReceiver);
```

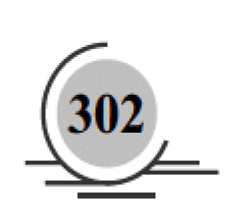
```
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        unbindService(serviceConnection);
        bleService = null;
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.gatt_services, menu);
        if (isConnected) {
            menu.findItem(R.id.menu_connect).setVisible(false);
            menu.findItem(R.id.menu_disconnect).setVisible(true);
        } else {
            menu.findItem(R.id.menu_connect).setVisible(true);
            menu.findItem(R.id.menu_disconnect).setVisible(false);
        }
        return true;
    }
//根据用户选择设置连接状态
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
            case R.id.menu_connect:
                bleService.connect(deviceAddress);
                return true;
            case R.id.menu_disconnect:
                bleService.disconnect();
                return true;
            case android.R.id.home:
                onBackPressed();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
//更新连接状态
    private void updateConnectionState(final int resourceId) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                connectionState.setText(resourceId);
            }
        });
    }
//显示数据
    private void displayData(String uuid, String data) {
         if (data != null) {
              if (uuid.equals(BleHeartRateSensor.getServiceUUIDString())) {
                  heartRateField.setText(data);
```

```
                } else {
                        dataField.setText(data);
                }
        }
}
 private boolean enableHeartRateSensor() {
        if (gattServiceAdapter == null)
                return false;
        final BluetoothGattCharacteristic characteristic = gattServiceAdapter
                        .getHeartRateCharacteristic();
        Log.d(TAG,"characteristic: " + characteristic);
        final BleSensor<?> sensor = BleSensors.getSensor(characteristic
                        .getService()
                        .getUuid()
                        .toString());
        if (heartRateSensor != null)
                bleService.enableSensor(heartRateSensor, false);
        if (sensor == null) {
                bleService.readCharacteristic(characteristic);
                return true;
        }
        if (sensor == heartRateSensor)
                return true;
        heartRateSensor = sensor;
        bleService.enableSensor(sensor, true);

      this.setServiceListener(demoClickListener);
        return true;
 }

private void displayGattServices(List<BluetoothGattService> gattServices) {
      if (gattServices == null)
          return;
      gattServiceAdapter = new BleServicesAdapter(this, gattServices);
      gattServiceAdapter.setServiceListener(demoClickListener);
      gattServicesList.setAdapter(gattServiceAdapter);
}
private static IntentFilter makeGattUpdateIntentFilter() {
      final IntentFilter intentFilter = new IntentFilter();
      intentFilter.addAction(BleService.ACTION_GATT_CONNECTED);
      intentFilter.addAction(BleService.ACTION_GATT_DISCONNECTED);
      intentFilter.addAction(BleService.ACTION_GATT_SERVICES_DISCOVERED);
      intentFilter.addAction(BleService.ACTION_DATA_AVAILABLE);
      return intentFilter;
}
}
```

在上述代码中，可以控制当前设备的连接状态，此功能是通过文件 BleService.java 实现的，具体实现代码如下所示。



```
/**
 *服务管理连接，并与托管给定的蓝牙 BLE 设备上的 GATT 服务器进行数据通信
 */
public class BleService extends Service {
    private final static String TAG = BleService.class.getSimpleName();

    private BluetoothManager bluetoothManager;
    private BluetoothAdapter adapter;
    private String deviceAddress;
    private BluetoothGatt gatt;
    private int connectionState = STATE_DISCONNECTED;

    private static final int STATE_DISCONNECTED = 0;
    private static final int STATE_CONNECTING = 1;
    private static final int STATE_CONNECTED = 2;

    private final static String INTENT_PREFIX = BleService.class.getPackage().getName();
    public final static String ACTION_GATT_CONNECTED = INTENT_PREFIX+".ACTION_GATT_CONNECTED";
    public final static String ACTION_GATT_DISCONNECTED = INTENT_PREFIX+".ACTION_GATT_
    DISCONNECTED";
    public final static String ACTION_GATT_SERVICES_DISCOVERED = INTENT_PREFIX+".ACTION_
    GATT_ SERVICES_DISCOVERED";
    public final static String ACTION_DATA_AVAILABLE = INTENT_PREFIX+".ACTION_DATA_AVAILABLE";
    public final static String EXTRA_SERVICE_UUID = INTENT_PREFIX+".EXTRA_SERVICE_UUID";
    public final static String EXTRA_CHARACTERISTIC_UUID = INTENT_PREFIX+".EXTRA_CHARA
    CTERISTIC_UUI";
    public final static String EXTRA_DATA = INTENT_PREFIX+".EXTRA_DATA";
    public final static String EXTRA_TEXT = INTENT_PREFIX+".EXTRA_TEXT";

    //实现回调方法，用于处理 GATT 事件的应用程序，例如，连接变化和发现服务
    private final BluetoothGattExecutor executor = new BluetoothGattExecutor() {

        @Override
        public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {
            super.onConnectionStateChange(gatt, status, newState);

            String intentAction;
            if (newState == BluetoothProfile.STATE_CONNECTED) {
                intentAction = ACTION_GATT_CONNECTED;
                connectionState = STATE_CONNECTED;
                broadcastUpdate(intentAction);
                Log.i(TAG, "Connected to GATT server.");
                // 连接成功后试图发现服务
                Log.i(TAG, "Attempting to start service discovery:" +
                        BleService.this.gatt.discoverServices());
            } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
                intentAction = ACTION_GATT_DISCONNECTED;
                connectionState = STATE_DISCONNECTED;
                Log.i(TAG, "Disconnected from GATT server.");
                broadcastUpdate(intentAction);
            }
```

```
        }

        @Override
        public void onServicesDiscovered(BluetoothGatt gatt, int status) {
            super.onServicesDiscovered(gatt, status);

            if (status == BluetoothGatt.GATT_SUCCESS) {
                broadcastUpdate(ACTION_GATT_SERVICES_DISCOVERED);
            } else {
                Log.w(TAG, "onServicesDiscovered received: " + status);
            }
        }

        @Override
        public void onCharacteristicRead(BluetoothGatt gatt,
                                         BluetoothGattCharacteristic characteristic,
                                         int status) {
            super.onCharacteristicRead(gatt, characteristic, status);

            if (status == BluetoothGatt.GATT_SUCCESS) {
                final BleSensor<?> sensor = BleSensors.getSensor(characteristic.getService().getUuid().toString());
                if (sensor != null) {
                    if (sensor.onCharacteristicRead(characteristic)) {
                        return;
                    }
                }

                broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic);
            }
        }

        @Override
        public void onCharacteristicChanged(BluetoothGatt gatt,
                                            BluetoothGattCharacteristic characteristic) {
            super.onCharacteristicChanged(gatt, characteristic);

            broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic);
        }
    };

    private void broadcastUpdate(final String action) {
        final Intent intent = new Intent(action);
        sendBroadcast(intent);
    }

    private void broadcastUpdate(final String action,
                                 final BluetoothGattCharacteristic characteristic) {
        final Intent intent = new Intent(action);
        intent.putExtra(EXTRA_SERVICE_UUID, characteristic.getService().getUuid().toString());
        intent.putExtra(EXTRA_CHARACTERISTIC_UUID, characteristic.getUuid().toString());
```

```
        final BleSensor<?> sensor = BleSensors.getSensor(characteristic.getService().getUuid().toString());
        if (sensor != null) {
            sensor.onCharacteristicChanged(characteristic);
            final String text = sensor.getDataString();
            intent.putExtra(EXTRA_TEXT, text);
            sendBroadcast(intent);
        } else {
            //对于其他的配置文件，写入十六进制格式的数据
            final byte[ ] data = characteristic.getValue();
            if (data != null && data.length > 0) {
                final StringBuilder stringBuilder = new StringBuilder(data.length);
                for (byte byteChar : data)
                    stringBuilder.append(String.format("%02X ", byteChar));
                intent.putExtra(EXTRA_TEXT, new String(data) + "\n" + stringBuilder.toString());
            }
        }
        sendBroadcast(intent);
    }

    public class LocalBinder extends Binder {
        public BleService getService() {
            return BleService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }

    @Override
    public boolean onUnbind(Intent intent) {
    //使用给定的设备后，你应该确保 BluetoothGatt.close()被调用，这样可以正确清理资源。在这个特定的例子
中，当 UI 是断开服务时调用 close()
        close();
        return super.onUnbind(intent);
    }

    /**
      *启用或禁用对一个给特性的通知
      *
      * @param sensor
      * @param enabled If true, enable notification.   False otherwise.
      */
     public void enableSensor(BleSensor<?> sensor, boolean enabled) {
         if (sensor == null)
             return;

         if (adapter == null || gatt == null) {
             Log.w(TAG, "BluetoothAdapter not initialized");
             return;
```

```
        }

        executor.enable(sensor, enabled);
        executor.execute(gatt);
    }

    private final IBinder mBinder = new LocalBinder();

   /**
    *初始化一个引用到本地蓝牙适配器
    *
    * @return Return true if the initialization is successful.
    */
   public boolean initialize() {
       // For API level 18 and above, get a reference to BluetoothAdapter through
       // BluetoothManager
       if (bluetoothManager == null) {
           bluetoothManager = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
           if (bluetoothManager == null) {
               Log.e(TAG, "Unable to initialize BluetoothManager.");
               return false;
           }
       }

       adapter = bluetoothManager.getAdapter();
       if (adapter == null) {
           Log.e(TAG, "Unable to obtain a BluetoothAdapter.");
           return false;
       }

       return true;
   }

   /**
    *连接到托管蓝牙 BLE 设备上的 GATT 服务器
    *
    * @param address The device address of the destination device.
    *
    * @return Return true if the connection is initiated successfully. The connection result
    *         is reported asynchronously through the
    *         {@code BluetoothGattCallback#onConnectionStateChange(android.bluetooth.BluetoothGatt, int, int)}
    *         callback.
    */
   public boolean connect(final String address) {
       if (adapter == null || address == null) {
           Log.w(TAG, "BluetoothAdapter not initialized or unspecified address.");
           return false;
       }

       //前面连接的设备。尝试重新连接
       if (deviceAddress != null && address.equals(deviceAddress)
```

```
                && gatt != null) {
            Log.d(TAG, "Trying to use an existing BluetoothGatt for connection.");
            if (gatt.connect()) {
                connectionState = STATE_CONNECTING;
                return true;
            } else {
                return false;
            }
        }

        final BluetoothDevice device = adapter.getRemoteDevice(address);
        if (device == null) {
            Log.w(TAG, "Device not found.   Unable to connect.");
            return false;
        }
        //因为要直接连接到设备上，所以我们设置了自动连接，参数设置为 false
        gatt = device.connectGatt(this, false, executor);
        Log.d(TAG, "Trying to create a new connection.");
        deviceAddress = address;
        connectionState = STATE_CONNECTING;
        return true;
    }

    /**
     *断开现有连接或取消一个挂起的连接。断线结果
     *通过异步
     * {@code BluetoothGattCallback#onConnectionStateChange(android.bluetooth.BluetoothGatt, int, int)}
     *  回调
     */
    public void disconnect() {
        if (adapter == null || gatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
        gatt.disconnect();
    }

    /**
     *使用给定的 BLE 设备后，应用程序必须调用此方法，以确保资源被正确释放
     */
    public void close() {
        if (gatt == null) {
            return;
        }
        gatt.close();
        gatt = null;
    }
    /**
     *读取一个给定的{@code BluetoothGattCharacteristic}. 读取结果通过异步
     *{@code BluetoothGattCallback#onCharacteristicRead(android.bluetooth.BluetoothGatt, android.bluetooth.
      Bluetooth GattCharacteristic, int)} 回调
```

```
     *
     * @param characteristic The characteristic to read from.
     */
    public void readCharacteristic(BluetoothGattCharacteristic characteristic) {
        if (adapter == null || gatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
        gatt.readCharacteristic(characteristic);
    }
    public void updateSensor(BleSensor<?> sensor) {
        if (sensor == null)
            return;
        if (adapter == null || gatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
        executor.update(sensor);
        executor.execute(gatt);
    }
    /**
     *检索连接的设备上支持关 GATT 服务列表
     * @return A {@code List} of supported services.
     */
    public List<BluetoothGattService> getSupportedGattServices() {
        if (gatt == null) return null;
        return gatt.getServices();
    }
}
```

在上述代码中，通过文件 BluetoothGattExecutor.java 实现了蓝牙 GATT 服务的具体操作，具体实现代码如下所示。

```
public class BluetoothGattExecutor extends BluetoothGattCallback {
    public interface ServiceAction {
        public static final ServiceAction NULL = new ServiceAction() {
            @Override
            public boolean execute(BluetoothGatt bluetoothGatt) {
                // it is null action. do nothing.
                return true;
            }
        };
        /***
         * 执行操作
         * @param bluetoothGatt
         * @return true - if action was executed instantly. false if action is waiting for
         *          feedback.
         */
        public boolean execute(BluetoothGatt bluetoothGatt);
    }
    private final LinkedList<BluetoothGattExecutor.ServiceAction> queue = new LinkedList<ServiceAction>();
    private volatile ServiceAction currentAction;
```

```
    public void update(final BleSensor sensor) {
        queue.add(sensor.update());
    }
    public void enable(BleSensor sensor, boolean enable) {
        final ServiceAction[ ] actions = sensor.enable(enable);
        for ( ServiceAction action : actions ) {
            this.queue.add(action);
        }
    }
    public void execute(BluetoothGatt gatt) {
        if (currentAction != null)
            return;
        boolean next = !queue.isEmpty();
        while (next) {
            final BluetoothGattExecutor.ServiceAction action = queue.pop();
            currentAction = action;
            if (!action.execute(gatt))
                break;
            currentAction = null;
            next = !queue.isEmpty();
        }
    }
    @Override
    public void onDescriptorWrite(BluetoothGatt gatt, BluetoothGattDescriptor descriptor, int status) {
        super.onDescriptorWrite(gatt, descriptor, status);
        currentAction = null;
        execute(gatt);
    }
    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
        currentAction = null;
        execute(gatt);
    }
    @Override
    public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {
        if (newState == BluetoothProfile.STATE_DISCONNECTED) {
            queue.clear();
        }
    }
    @Override
    public void onCharacteristicRead(BluetoothGatt gatt,
                                     BluetoothGattCharacteristic characteristic,
                                     int status) {
        currentAction = null;
        execute(gatt);
    }
}
```

蓝牙设置界面执行效果如图 11-2 所示。

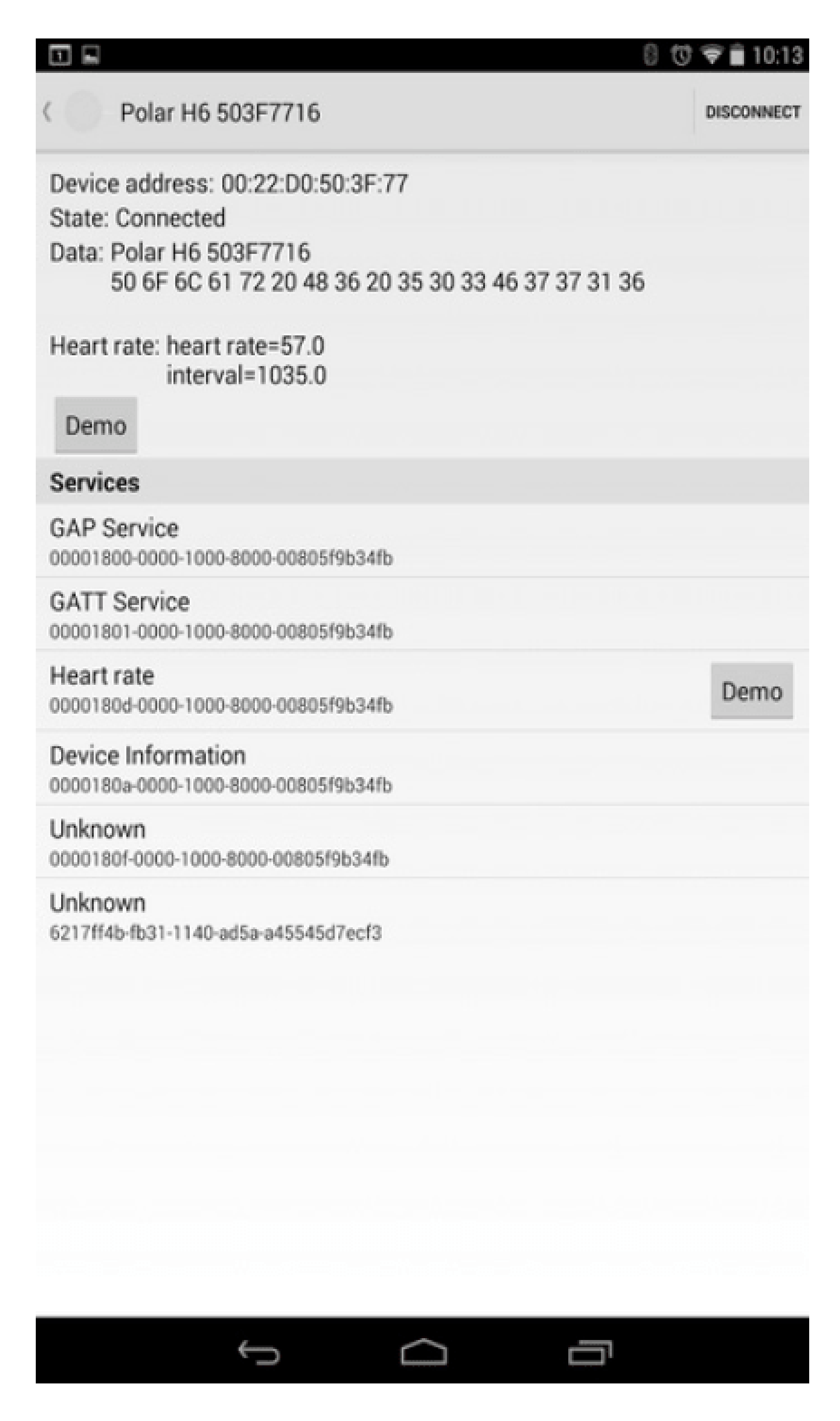


图 11-2　某个蓝牙设备的控制界面

11.2.3　蓝牙 BLE 设备适配器

在前面 11.2.2 的内容中用到了蓝牙 BLE 设备适配器功能，通过适配器列表显示了当前扫描到的蓝牙 BLE 设备。蓝牙 BLE 设备适配器的布局文件是 listitem_device.xml，具体实现代码如下所示。

```
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="4dp"
        android:paddingBottom="4dp"
        android:paddingRight="10dp"
        android:paddingLeft="10dp">
    <TextView
            android:id="@+id/device_rssi"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"/>

    <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/device_rssi">
```

```
        <TextView
                android:id="@+id/device_name"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAppearance="?android:textAppearanceMedium"
                android:textStyle="bold"/>
        <TextView
                android:id="@+id/device_address"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"/>
    </LinearLayout>
</RelativeLayout>
```

对应的程序文件是 BleDevicesAdapter.java，功能是扫描附近的 BLE 蓝牙设备，具体实现代码如下所示。

```
public class BleDevicesAdapter extends BaseAdapter {
    private final LayoutInflater inflater;

    private final ArrayList<BluetoothDevice> leDevices;
    private final HashMap<BluetoothDevice, Integer> rssiMap = new HashMap<BluetoothDevice, Integer>();

    public BleDevicesAdapter(Context context) {
        leDevices = new ArrayList<BluetoothDevice>();
        inflater = LayoutInflater.from(context);
    }

    public void addDevice(BluetoothDevice device, int rssi) {
        if (!leDevices.contains(device)) {
            leDevices.add(device);
        }
        rssiMap.put(device, rssi);
    }

    public BluetoothDevice getDevice(int position) {
        return leDevices.get(position);
    }

    public void clear() {
        leDevices.clear();
    }

    @Override
    public int getCount() {
        return leDevices.size();
    }

    @Override
    public Object getItem(int i) {
        return leDevices.get(i);
    }
```

```
    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        ViewHolder viewHolder;
        //一般列表视图的优化代码
        if (view == null) {
            view = inflater.inflate(R.layout.listitem_device, null);
            viewHolder = new ViewHolder();
            viewHolder.deviceAddress = (TextView) view.findViewById(R.id.device_address);
            viewHolder.deviceName = (TextView) view.findViewById(R.id.device_name);
            viewHolder.deviceRssi = (TextView) view.findViewById(R.id.device_rssi);
            view.setTag(viewHolder);
        } else {
            viewHolder = (ViewHolder) view.getTag();
        }

        BluetoothDevice device = leDevices.get(i);
        final String deviceName = device.getName();
        if (deviceName != null && deviceName.length() > 0)
            viewHolder.deviceName.setText(deviceName);
        else
            viewHolder.deviceName.setText(R.string.unknown_device);
        viewHolder.deviceAddress.setText(device.getAddress());
        viewHolder.deviceRssi.setText(""+rssiMap.get(device)+" dBm");

        return view;
    }

    private static class ViewHolder {
        TextView deviceName;
        TextView deviceAddress;
        TextView deviceRssi;
    }
}
```

11.2.4 蓝牙 BLE 服务适配器

在前面 11.2.2 的内容中用到了蓝牙 BLE 服务适配器功能，通过适配器列表显示了当前扫描到的蓝牙 BLE 服务。蓝牙 BLE 服务适配器的布局文件是 listitem_service.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="4dp"
        android:paddingBottom="4dp"
```

```
        android:paddingRight="10dp"
        android:paddingLeft="10dp">

    <Button
            android:id="@+id/demo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"
            android:text="@string/action_demo"
            android:clickable="false"
            android:focusable="false" />
    <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/modes"
            android:textAppearance="?android:textAppearanceMedium" />
    <TextView
            android:id="@+id/uuid"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/modes"
            android:layout_below="@+id/name" />

</RelativeLayout>
```

对应的程序文件是 BleServicesAdapter.java，具体实现代码如下所示。

```
public class BleServicesAdapter extends BaseExpandableListAdapter {

    private final static String TAG = BleServicesAdapter.class.getSimpleName();

    public interface OnServiceItemClickListener {
        public void onDemoClick(BluetoothGattService service);

        public void onServiceEnabled(BluetoothGattService service,
                boolean enabled);

        public void onServiceUpdated(BluetoothGattService service);
    }

    private static final String MODE_READ = "R";
    private static final String MODE_NOTIFY = "N";
    private static final String MODE_WRITE = "W";

    private final ArrayList<BluetoothGattService> services;
    private final HashMap<BluetoothGattService, ArrayList<BluetoothGattCharacteristic>> characteristics;
    private final LayoutInflater inflater;

    private BluetoothGattService heartRateService;
    private BluetoothGattCharacteristic heartRateCharacteristic;
```

```
private OnServiceItemClickListener serviceListener;

public BleServicesAdapter(Context context,
        List<BluetoothGattService> gattServices) {
    inflater = LayoutInflater.from(context);

    services = new ArrayList<BluetoothGattService>(gattServices.size());
    characteristics = new HashMap<BluetoothGattService, ArrayList<BluetoothGattCharacteristic>>(
            gattServices.size());
    for (BluetoothGattService gattService : gattServices) {
        final List<BluetoothGattCharacteristic> gattCharacteristics = gattService
                .getCharacteristics();
        characteristics.put(gattService,
                new ArrayList<BluetoothGattCharacteristic>(
                        gattCharacteristics));
        services.add(gattService);

        if (gattService.getUuid().equals(
                UUID.fromString(BleHeartRateSensor.getServiceUUIDString()))) {
            heartRateService = gattService;
            heartRateCharacteristic = gattCharacteristics.get(0);
        }
    }
}

public void setServiceListener(OnServiceItemClickListener listener) {
    this.serviceListener = listener;
}

@Override
public int getGroupCount() {
    return services.size();
}

@Override
public int getChildrenCount(int groupPosition) {
    return characteristics.get(getGroup(groupPosition)).size();
}

@Override
public BluetoothGattService getGroup(int groupPosition) {
    return services.get(groupPosition);
}

@Override
public BluetoothGattCharacteristic getChild(int groupPosition,
        int childPosition) {
    Log.d(TAG, "group:" + groupPosition + " child:" + childPosition);
    Log.d(TAG,
            "uuid:"
```

```
                        + characteristics.get(getGroup(groupPosition))
                                .get(childPosition).getUuid());
        return characteristics.get(getGroup(groupPosition)).get(childPosition);
    }

    public BluetoothGattService getHeartRateService() {
        return heartRateService;
    }

    public BluetoothGattCharacteristic getHeartRateCharacteristic() {
        return heartRateCharacteristic;
    }

    @Override
    public long getGroupId(int groupPosition) {
        return groupPosition;
    }

    @Override
    public long getChildId(int groupPosition, int childPosition) {
        return groupPosition * 100 + childPosition;
    }

    @Override
    public boolean hasStableIds() {
        return true;
    }

    @Override
    public View getGroupView(int groupPosition, boolean isExpanded,
            View convertView, ViewGroup parent) {
        final GroupViewHolder holder;
        if (convertView == null) {
            holder = new GroupViewHolder();

            convertView = inflater.inflate(R.layout.listitem_service, parent,
                    false);
            holder.name = (TextView) convertView.findViewById(R.id.name);
            holder.uuid = (TextView) convertView.findViewById(R.id.uuid);
            holder.demo = convertView.findViewById(R.id.demo);

            holder.demo.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (serviceListener == null)
                        return;
                    final BluetoothGattService service = (BluetoothGattService) holder.demo
                            .getTag();
                    serviceListener.onDemoClick(service);
                }
            });
```

```
                convertView.setTag(holder);
            } else {
                holder = (GroupViewHolder) convertView.getTag();
            }

            final BluetoothGattService item = getGroup(groupPosition);

            final String uuid = item.getUuid().toString();
            final BleSensor<?> sensor = BleSensors.getSensor(uuid);
            final BleInfoService infoService = BleInfoServices.getService(uuid);

            final String serviceName;

            if (sensor != null)
                serviceName = sensor.getName();
            else if (infoService != null)
                serviceName = infoService.getName();
            else
                serviceName = "Unknown";

            holder.name.setText(serviceName);
            holder.uuid.setText(uuid);
            if (isDemoable(sensor)) {
                holder.demo.setTag(item);
                holder.demo.setVisibility(View.VISIBLE);
            } else {
                holder.demo.setVisibility(View.GONE);
            }

            return convertView;
        }

        @Override
        public View getChildView(int groupPosition, int childPosition,
                boolean isLastChild, View convertView, ViewGroup parent) {
            final ChildViewHolder holder;
            if (convertView == null) {
                holder = new ChildViewHolder();

                convertView = inflater.inflate(R.layout.listitem_characteristic,
                        parent, false);
                holder.name = (TextView) convertView.findViewById(R.id.name);
                holder.uuid = (TextView) convertView.findViewById(R.id.uuid);
                holder.modes = (TextView) convertView.findViewById(R.id.modes);
                holder.seek = (SeekBar) convertView.findViewById(R.id.seek);
                holder.seek
                        .setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
                            @Override
                            public void onProgressChanged(SeekBar seekBar,
                                    int progress, boolean fromUser) {
```

```
                if (serviceListener == null || !fromUser)
                    return;

                final BleSensor<?> sensor = BleSensors
                        .getSensor(holder.service.getUuid()
                                .toString());
                if (sensor == null)
                    return;

            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });

    convertView.setTag(holder);
} else {
    holder = (ChildViewHolder) convertView.getTag();
}

final BluetoothGattCharacteristic item = getChild(groupPosition,
        childPosition);

final String uuid = item.getUuid().toString();
final String name;
final String modes = getModeString(item.getProperties());

holder.service = item.getService();

final String serviceUUID = item.getService().getUuid().toString();
final BleSensor<?> sensor = BleSensors.getSensor(serviceUUID);
final BleInfoService infoService = BleInfoServices
        .getService(serviceUUID);

if (sensor != null) {
    name = sensor.getCharacteristicName(uuid);

    if (sensor.isConfigUUID(uuid)) {

    } else {
        holder.uuid.setVisibility(View.VISIBLE);
        holder.seek.setVisibility(View.GONE);
    }
} else if (infoService != null) {
    name = infoService.getCharacteristicName(uuid);
```

```
                holder.uuid.setVisibility(View.VISIBLE);
                holder.seek.setVisibility(View.GONE);
            } else {
                name = "Unknown";

                holder.uuid.setVisibility(View.VISIBLE);
                holder.seek.setVisibility(View.GONE);
            }

            holder.name.setText(name);
            holder.uuid.setText(uuid);
            holder.modes.setText(modes);

            return convertView;
        }

        @Override
        public boolean isChildSelectable(int groupPosition, int childPosition) {
            return true;
        }

        private static boolean isDemoable(BleSensor<?> sensor) {
            if (sensor instanceof BleHeartRateSensor)
                return true;
            return false;
        }

        private static String getModeString(int prop) {
            final StringBuilder modeBuilder = new StringBuilder();
            if ((prop & BluetoothGattCharacteristic.PROPERTY_READ) > 0) {
                modeBuilder.append(MODE_READ);
            }
            if ((prop & BluetoothGattCharacteristic.PROPERTY_NOTIFY) > 0) {
                if (modeBuilder.length() > 0)
                    modeBuilder.append("/");
                modeBuilder.append(MODE_NOTIFY);
            }
            if ((prop & BluetoothGattCharacteristic.PROPERTY_WRITE) > 0) {
                if (modeBuilder.length() > 0)
                    modeBuilder.append("/");
                modeBuilder.append(MODE_WRITE);
            }
            return modeBuilder.toString();
        }

        private static class GroupViewHolder {
            public TextView name;
            public TextView uuid;
            public View demo;
        }
```

```
    private static class ChildViewHolder {
        public BluetoothGattService service;

        public TextView name;
        public TextView uuid;
        public TextView modes;
        public SeekBar seek;
    }
}
```

11.2.5　传感器测试心率

本实例的核心功能是通过传感器测试心率，此功能的核心文件是 BleHeartRateSensor.java，功能是获取当前蓝牙设备的信息参数，例如服务 UUID、数据 UUID 等，并输出获得心率的信息，输出的信息有 UINT16 和 UINT8 两种格式。

```
public class BleHeartRateSensor extends BleSensor<float[ ]> {
    private final static String TAG = BleHeartRateSensor.class.getSimpleName();
    private static final String UUID_SENSOR_BODY_LOCATION = "00002a38-0000-1000-8000-00805f9b34fb";
    private static final int SENSOR_BODY_LOCATION_OTHER = 0;
    private static final int SENSOR_BODY_LOCATION_CHEST = 1;
    private static final int SENSOR_BODY_LOCATION_WRIST = 2;
    private static final int SENSOR_BODY_LOCATION_FINGER = 3;
    private static final int SENSOR_BODY_LOCATION_HAND = 4;
    private static final int SENSOR_BODY_LOCATION_EAR = 5;
    private static final int SENSOR_BODY_LOCATION_FOOT = 6;
    private int location = -1;
    BleHeartRateSensor() {
        super();
    }
    @Override
    public String getName() {
        return "Heart rate";
    }
    @Override
    public String getServiceUUID() {
        return "0000180d-0000-1000-8000-00805f9b34fb";
    }

    public static String getServiceUUIDString() {
        return "0000180d-0000-1000-8000-00805f9b34fb";
    }

    @Override
    public String getDataUUID() {
        return "00002a37-0000-1000-8000-00805f9b34fb";
    }

    public static String getDataUUIDString() {
        return "00002a37-0000-1000-8000-00805f9b34fb";
```

```
    }

    @Override
    public String getConfigUUID() {
        return "00002902-0000-1000-8000-00805f9b34fb";
    }

    @Override
    public String getCharacteristicName(String uuid) {
        if (UUID_SENSOR_BODY_LOCATION.equals(uuid))
            return getName() + " Sensor body location";
        return super.getCharacteristicName(uuid);
    }

    @Override
    public boolean onCharacteristicRead(BluetoothGattCharacteristic c) {
        super.onCharacteristicRead(c);
        Log.d(TAG, "onCharacteristicsReas");

        if ( !c.getUuid().toString().equals(UUID_SENSOR_BODY_LOCATION) )
            return false;
        location = c.getProperties();
        Log.d(TAG, "Sensor body location: " + location);
        return true;
    }
    @Override
    public String getDataString() {
        final float[ ] data = getData();
        return "heart rate=" + data[0] + "\ninterval=" + data[1];
    }

    @Override
    public float[ ] parse(BluetoothGattCharacteristic c) {

        double heartRate = extractHeartRate(c);
        double contact = extractContact(c);
        double energy = extractEnergyExpended(c);
        Integer[ ] interval = extractBeatToBeatInterval(c);

        float[ ] result = null;
        if (interval != null) {
            result = new float[interval.length + 1];
        } else {
            result = new float[2];
            result[1] = -1.0f;
        }
        result[0] = (float) heartRate;

        if (interval != null) {
            for (int i = 0; i < interval.length; i++) {
                result[i+1] = interval[i].floatValue();
```

```
            }
        }

        return result;
    }

    private static double extractHeartRate(
            BluetoothGattCharacteristic characteristic) {

        int flag = characteristic.getProperties();
        Log.d(TAG, "Heart rate flag: " + flag);
        int format = -1;
        //心率数字格式
        if ((flag & 0x01) != 0) {
            format = BluetoothGattCharacteristic.FORMAT_UINT16;
            Log.d(TAG, "Heart rate format UINT16.");
        } else {
            format = BluetoothGattCharacteristic.FORMAT_UINT8;
            Log.d(TAG, "Heart rate format UINT8.");
        }
        final int heartRate = characteristic.getIntValue(format, 1);
        Log.d(TAG, String.format("Received heart rate: %d", heartRate));
        return heartRate;
    }

    private static double extractContact(
            BluetoothGattCharacteristic characteristic) {

        int flag = characteristic.getProperties();
        int format = -1;
        //传感器连接状态
        if ((flag & 0x02) != 0) {
            Log.d(TAG, "Heart rate sensor contact info exists");
            if ((flag & 0x04) != 0) {
                Log.d(TAG, "Heart rate sensor contact is ON");
            } else {
                Log.d(TAG, "Heart rate sensor contact is OFF");
            }
        } else{
            Log.d(TAG, "Heart rate sensor contact info doesn't exists");
        }
        //final int heartRate = characteristic.getIntValue(format, 1);
        //Log.d(TAG, String.format("Received heart rate: %d", heartRate));
        return 0.0d;
    }

    private static double extractEnergyExpended(
            BluetoothGattCharacteristic characteristic) {

        int flag = characteristic.getProperties();
        int format = -1;
```

```
        // 计算活力状态
        if ((flag & 0x08) != 0) {
            Log.d(TAG, "Heart rate energy calculation exists.");
        } else {
            Log.d(TAG, "Heart rate energy calculation doesn't exists.");
        }
        //final int heartRate = characteristic.getIntValue(format, 1);
        //Log.d(TAG, String.format("Received heart rate: %d", heartRate));
        return 0.0d;
    }

    private static Integer[ ] extractBeatToBeatInterval(
                BluetoothGattCharacteristic characteristic) {

        int flag = characteristic.getIntValue(BluetoothGattCharacteristic.FORMAT_UINT8, 0);
        int format = -1;
        int energy = -1;
        int offset = 1; // This depends on hear rate value format and if there is energy data
        int rr_count = 0;

        if ((flag & 0x01) != 0) {
            format = BluetoothGattCharacteristic.FORMAT_UINT16;
            Log.d(TAG, "Heart rate format UINT16.");
            offset = 3;
        } else {
            format = BluetoothGattCharacteristic.FORMAT_UINT8;
            Log.d(TAG, "Heart rate format UINT8.");
            offset = 2;
        }
        if ((flag & 0x08) != 0) {
            //目前卡路里
            energy = characteristic.getIntValue(BluetoothGattCharacteristic.FORMAT_UINT16, offset);
            offset += 2;
            Log.d(TAG, "Received energy: { }"+ energy);
        }
        if ((flag & 0x16) != 0){
            // stuff 值
            Log.d(TAG, "RR stuff found at offset: "+ offset);
            Log.d(TAG, "RR length: "+ (characteristic.getValue()).length);
            rr_count = ((characteristic.getValue()).length - offset) / 2;
            Log.d(TAG, "RR length: "+ (characteristic.getValue()).length);
            Log.d(TAG, "rr_count: "+ rr_count);
              if (rr_count > 0) {
                  Integer[ ] mRr_values = new Integer[rr_count];
                  for (int i = 0; i < rr_count; i++) {
                      mRr_values[i] = characteristic.getIntValue(
                              BluetoothGattCharacteristic.FORMAT_UINT16, offset);
                      offset += 2;
                      Log.d(TAG, "Received RR: " + mRr_values[i]);
                  }
                  return mRr_values;
```

```
                }
            }
            Log.d(TAG, "No RR data on this update: ");
            return null;
        }
}
```

11.2.6　图形化显示心率值

当读取到传感器的心率值后，会通过 BLE 蓝牙将数据传输到检测设备中，在 Android 系统中通过图形化界面方式显示检测到的心率值。上述功能的 UI 布局文件是 demo_opengl.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    <com.sample.hrv.demo.DemoGLSurfaceView
            android:id="@+id/gl"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    <TextView
            android:id="@+id/text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:textAppearance="?android:textAppearanceMedium"/>
</FrameLayout>
```

对应的程序文件是 DemoHeartRateSensorActivity.java，功能是根据获取的心率值构建一个图形化的心率图，具体实现代码如下所示。

```
public class DemoHeartRateSensorActivity extends DemoSensorActivity {
    private final static String TAG = DemoHeartRateSensorActivity.class
            .getSimpleName();

    private TextView viewText;
    private PolygonRenderer renderer;

    private GLSurfaceView view;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.demo_opengl);
        view = (GLSurfaceView) findViewById(R.id.gl);

        getActionBar().setTitle(R.string.title_demo_heartrate);
```

```
        viewText = (TextView) findViewById(R.id.text);

        renderer = new PolygonRenderer(this);
        view.setRenderer(renderer);
        //view.setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
        // Render when hear rate data is updated
        view.setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);
    }

    @Override
    public void onDataRecieved(BleSensor<?> sensor, String text) {
        if (sensor instanceof BleHeartRateSensor) {
            final BleHeartRateSensor heartSensor = (BleHeartRateSensor) sensor;
            float[ ] values = heartSensor.getData();
            renderer.setInterval(values);
            view.requestRender();

            viewText.setText(text);
        }
    }

    public abstract class AbstractRenderer implements GLSurfaceView.Renderer {

        public int[ ] getConfigSpec() {
            int[ ] configSpec = { EGL10.EGL_DEPTH_SIZE, 0, EGL10.EGL_NONE };
            return configSpec;
        }

        public void onSurfaceCreated(GL10 gl, EGLConfig eglConfig) {
            gl.glDisable(GL10.GL_DITHER);
            gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_FASTEST);
            gl.glClearColor(.5f, .5f, .5f, 1);
            gl.glShadeModel(GL10.GL_SMOOTH);
            gl.glEnable(GL10.GL_DEPTH_TEST);
        }

        public void onSurfaceChanged(GL10 gl, int w, int h) {
            gl.glViewport(0, 0, w, h);
            float ratio = (float) w / h;
            gl.glMatrixMode(GL10.GL_PROJECTION);
            gl.glLoadIdentity();
            gl.glFrustumf(-ratio, ratio, -1, 1, 3, 7);
        }

        public void onDrawFrame(GL10 gl) {
            gl.glDisable(GL10.GL_DITHER);
            gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
            gl.glMatrixMode(GL10.GL_MODELVIEW);
            gl.glLoadIdentity();
            GLU.gluLookAt(gl, 0, 0, -5, 0f, 0f, 0f, 0f, 1.0f, 0.0f);
```

```
            gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
            draw(gl);
        }

        protected abstract void draw(GL10 gl);
}

public class PolygonRenderer extends AbstractRenderer {
        private final String TAG = PolygonRenderer.class
                    .getSimpleName();

        // Number of points or vertices we want to use
        private final static int VERTS = 4;

        // A raw native buffer to hold the point coordinates
        private FloatBuffer mFVertexBuffer;

        // A raw native buffer to hold indices
        // allowing a reuse of points.
        private ShortBuffer mIndexBuffer;

        private int numOfIndecies = 0;

        private long prevtime = SystemClock.uptimeMillis();

        private int sides = 32;

        private float[ ] interval = { 0, 0, 0 };
        private float previousInterval = 0;

        public void setInterval(float[ ] interval) {
            if (this.interval[1] >= 0 && interval[1] > 0) {
                    this.previousInterval = this.interval[1];
            }
            this.interval[0] = interval[0]; // heart rate
            this.interval[1] = interval[1]; // beat to beat interval
            this.interval[2] = 0;                    // empty
        }

        public PolygonRenderer(Context context) {
            prepareBuffers(sides, interval[1]);
        }

        private void prepareBuffers(int sides, float radius) {
            Log.d(TAG,"radius: "+radius +" previous: "+previousInterval);
            // Is it a valid value?
            if (radius < 0) {
                    radius = previousInterval;
            }
```

```
            // Double check if the previous value was valid
            if (radius < 0) {
                radius = 700;
            }
            Log.d(TAG,"final radius: "+radius);

            radius = ( ( radius / 1000 ) - 0.7f ) * 2;

            RegularPolygon t = new RegularPolygon(0, 0, 0, radius, sides);
            this.mFVertexBuffer = t.getVertexBuffer();
            this.mIndexBuffer = t.getIndexBuffer();
            this.numOfIndecies = t.getNumberOfIndecies();
            this.mFVertexBuffer.position(0);
            this.mIndexBuffer.position(0);
        }
        // overriden method
        protected void draw(GL10 gl) {
            long curtime = SystemClock.uptimeMillis();
            this.prepareBuffers(sides, interval[1]);
            gl.glColor4f(96/255.0f, 246/255.0f, 255/255.0f, 1.0f);
            gl.glVertexPointer(3, GL10.GL_FLOAT, 0, mFVertexBuffer);
            gl.glDrawElements(GL10.GL_TRIANGLES, this.numOfIndecies,
                    GL10.GL_UNSIGNED_SHORT, mIndexBuffer);
        }
    }
    private static class RegularPolygon {
        private static final String TAG = RegularPolygon.class
                .getSimpleName();
        private float cx, cy, cz, r;
        private int sides;
        private float[ ] xarray = null;
        private float[ ] yarray = null;
        public RegularPolygon(float incx, float incy, float incz, // (x,y,z)
                                                                      // center
                float inr, // radius
                int insides) // number of sides
        {
            cx = incx;
            cy = incy;
            cz = incz;
            r = inr;
            sides = insides;
            xarray = new float[sides];
            yarray = new float[sides];
            calcArrays();
        }
        private void calcArrays() {
            float[ ] xmarray = this.getXMultiplierArray();
            float[ ] ymarray = this.getYMultiplierArray();
            // calc xarray
```

```
        for (int i = 0; i < sides; i++) {
            float curm = xmarray[i];
            float xcoord = cx + r * curm;
            xarray[i] = xcoord;
        }
        //this.printArray(xarray, "xarray");
        // calc yarray
        for (int i = 0; i < sides; i++) {
            float curm = ymarray[i];
            float ycoord = cy + r * curm;
            yarray[i] = ycoord;
        }
        //this.printArray(yarray, "yarray");
    }
    public FloatBuffer getVertexBuffer() {
        int vertices = sides + 1;
        int coordinates = 3;
        int floatsize = 4;
        int spacePerVertex = coordinates * floatsize;

        ByteBuffer vbb = ByteBuffer.allocateDirect(spacePerVertex
                * vertices);
        vbb.order(ByteOrder.nativeOrder());
        FloatBuffer mFVertexBuffer = vbb.asFloatBuffer();

        // Put the first coordinate (x,y,z:0,0,0)
        mFVertexBuffer.put(0.0f); // x
        mFVertexBuffer.put(0.0f); // y
        mFVertexBuffer.put(0.0f); // z

        int totalPuts = 3;
        for (int i = 0; i < sides; i++) {
            mFVertexBuffer.put(xarray[i]); // x
            mFVertexBuffer.put(yarray[i]); // y
            mFVertexBuffer.put(0.0f); // z
            totalPuts += 3;
        }
        //Log.d(TAG, "total puts: " + Integer.toString(totalPuts));
        return mFVertexBuffer;
    }
    public ShortBuffer getIndexBuffer() {
        short[ ] iarray = new short[sides * 3];
        ByteBuffer ibb = ByteBuffer.allocateDirect(sides * 3 * 2);
        ibb.order(ByteOrder.nativeOrder());
        ShortBuffer mIndexBuffer = ibb.asShortBuffer();
        for (int i = 0; i < sides; i++) {
            short index1 = 0;
            short index2 = (short) (i + 1);
            short index3 = (short) (i + 2);
            if (index3 == sides + 1) {
```

```
                index3 = 1;
            }
            mIndexBuffer.put(index1);
            mIndexBuffer.put(index2);
            mIndexBuffer.put(index3);

            iarray[i * 3 + 0] = index1;
            iarray[i * 3 + 1] = index2;
            iarray[i * 3 + 2] = index3;
        }
        //this.printShortArray(iarray, "index array");
        return mIndexBuffer;
    }
    private float[ ] getXMultiplierArray() {
        float[ ] angleArray = getAngleArrays();
        float[ ] xmultiplierArray = new float[sides];
        for (int i = 0; i < angleArray.length; i++) {
            float curAngle = angleArray[i];
            float sinvalue = (float) Math.cos(Math.toRadians(curAngle));
            float absSinValue = Math.abs(sinvalue);
            if (isXPositiveQuadrant(curAngle)) {
                sinvalue = absSinValue;
            } else {
                sinvalue = -absSinValue;
            }
            xmultiplierArray[i] = this.getApproxValue(sinvalue);
        }
        //this.printArray(xmultiplierArray, "xmultiplierArray");
        return xmultiplierArray;
    }
    private float[ ] getYMultiplierArray() {
        float[ ] angleArray = getAngleArrays();
        float[ ] ymultiplierArray = new float[sides];
        for (int i = 0; i < angleArray.length; i++) {
            float curAngle = angleArray[i];
            float sinvalue = (float) Math.sin(Math.toRadians(curAngle));
            float absSinValue = Math.abs(sinvalue);
            if (isYPositiveQuadrant(curAngle)) {
                sinvalue = absSinValue;
            } else {
                sinvalue = -absSinValue;
            }
            ymultiplierArray[i] = this.getApproxValue(sinvalue);
        }
        //this.printArray(ymultiplierArray, "ymultiplierArray");
        return ymultiplierArray;
    }
    private boolean isXPositiveQuadrant(float angle) {
        if ((0 <= angle) && (angle <= 90)) {
            return true;
```

```
        }
        if ((angle < 0) && (angle >= -90)) {
            return true;
        }
        return false;
    }
    private boolean isYPositiveQuadrant(float angle) {
        if ((0 <= angle) && (angle <= 90)) {
            return true;
        }
        if ((angle < 180) && (angle >= 90)) {
            return true;
        }
        return false;
    }
    private float[ ] getAngleArrays() {
        float[ ] angleArray = new float[sides];
        float commonAngle = 360.0f / sides;
        float halfAngle = commonAngle / 2.0f;
        float firstAngle = 360.0f - (90 + halfAngle);
        angleArray[0] = firstAngle;
        float curAngle = firstAngle;
        for (int i = 1; i < sides; i++) {
            float newAngle = curAngle - commonAngle;
            angleArray[i] = newAngle;
            curAngle = newAngle;
        }
        //printArray(angleArray, "angleArray");
        return angleArray;
    }
    private float getApproxValue(float f) {
        if (Math.abs(f) < 0.001) {
            return 0;
        }
        return f;
    }
    public int getNumberOfIndecies() {
        return sides * 3;
    }
    private void printArray(float array[ ], String tag) {
        StringBuilder sb = new StringBuilder(tag);
        for (int i = 0; i < array.length; i++) {
            sb.append(";").append(array[i]);
        }
        Log.d(TAG, sb.toString());
    }
    private void printShortArray(short array[ ], String tag) {
        StringBuilder sb = new StringBuilder(tag);
        for (int i = 0; i < array.length; i++) {
            sb.append(";").append(array[i]);
```

```
                }
                Log.d(TAG, sb.toString());
            }
        }
}
```

执行效果如图 11-3 所示。

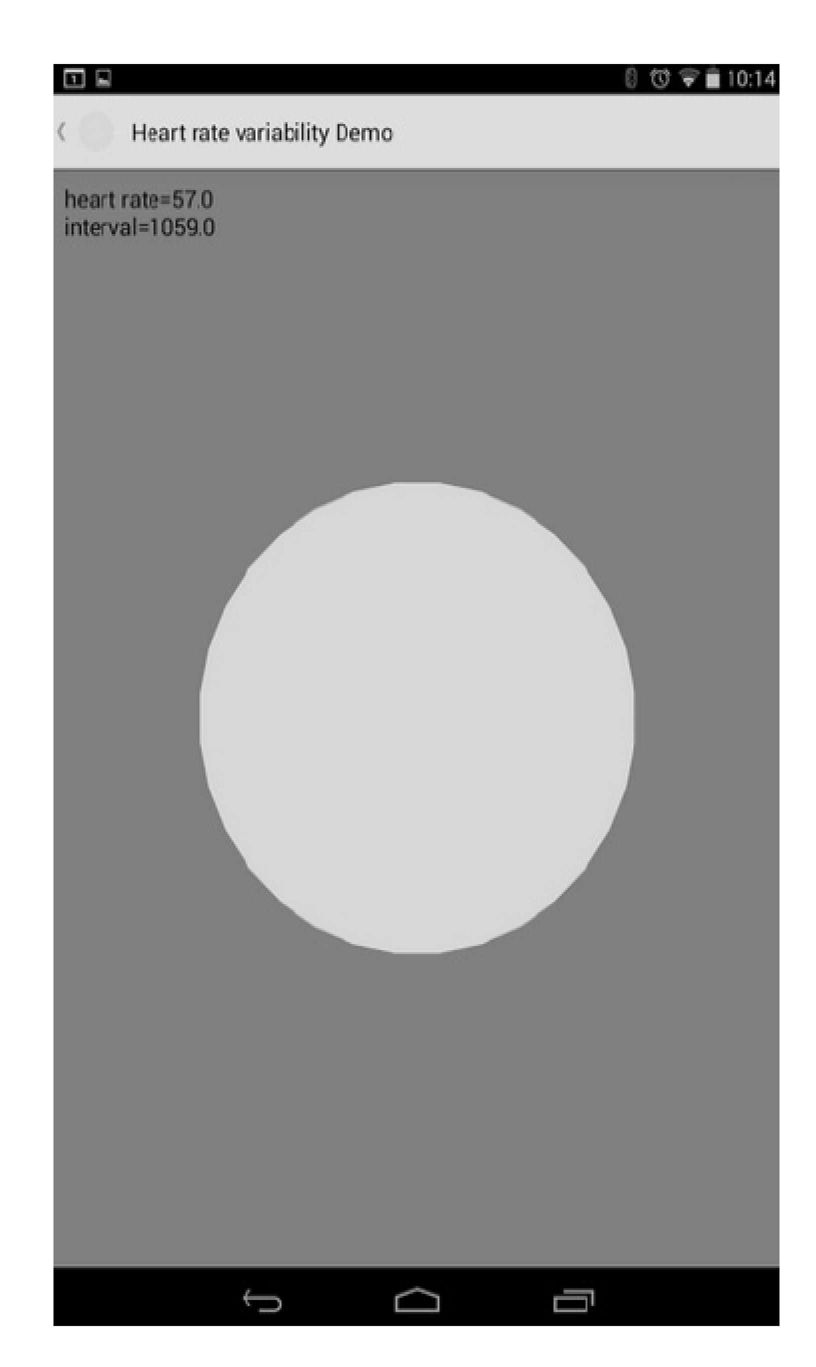


图 11-3　图形化心率界面

第 12 章 湿度测试仪

和测量重量、温度一样，选择湿度传感器首先要确定测量范围。除了气象、科研部门外，高温、湿度测控一般不需要全湿程（0～100%RH）测量。在当今的信息时代，传感器技术与计算机技术、自动控制技术紧密结合着。测量的目的在于控制，测量范围与控制范围合称使用范围。当然，对不需要搞测控系统的应用者来说，直接选择通用型湿度仪就可以了。本章将详细讲解 Android 设备中湿度传感器的基本知识，为读者学习本书后面的知识打下基础。

12.1 实现主界面

知识点讲解：光盘:视频\知识点\第 12 章\实现主界面.avi

本章将通过一个演示实例来详细讲解使用湿度传感器开发一个湿度测试仪的方法，本实例可以在 Android 设备上运行，测试当前天气环境下的湿度。在本节的内容中，将首先讲解实现主界面的具体过程。

实例	功能	源码路径
实例 12-1	获取远程湿度传感器的数据	光盘:\daima\12\humiditytrackerEX

12.1.1 实现主界面布局文件

编写主界面布局文件 main.xml，功能是通过按钮控件在主界面中分别显示 6 监测点的值，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:padding="10dp"
     >
<Button
      android:id="@+id/reading"
      android:text="@string/takereading"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_weight="0"
      style="@style/BigFont"
      android:paddingTop="15dp"
      android:paddingBottom="15dp"
      />
<TableLayout
      android:layout_width="fill_parent"
```

```
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:stretchColumns="0,1"
        >
        <TableRow>
            <Button
                android:id="@+id/stored1"
                android:textColor="@color/stored1"
                style="@style/StoredButton"
                />
            <Button
                android:id="@+id/stored2"
                android:textColor="@color/stored2"
                style="@style/StoredButton"
                />
        </TableRow>
        <TableRow>
            <Button
                android:id="@+id/stored3"
                android:textColor="@color/stored3"
                style="@style/StoredButton"
                />
            <Button
                android:id="@+id/stored4"
                android:textColor="@color/stored4"
                style="@style/StoredButton"
                />
        </TableRow>
        <TableRow>
            <Button
                android:id="@+id/stored5"
                android:textColor="@color/stored5"
                style="@style/StoredButton"
                />
            <Button
                android:id="@+id/stored6"
                android:textColor="@color/stored6"
                style="@style/StoredButton"
                />
        </TableRow>
    </TableLayout>

    <TextView
        android:text="@string/copyright"
        android:textSize="12dp"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />

</LinearLayout>
```

12.1.2　主 Activity 的实现文件

主 Activity 的实现文件是 HumidityActivity.java，功能是监听用户触摸单击了屏幕中的哪一个按钮，并执行对应的处理事件函数。文件 HumidityActivity.java 的具体实现代码如下所示。

```
public class HumidityActivity extends Activity {
	public static final String EXTRA_HUMIDITY = "com.leafdigital.humidity.Humidity";
	public static final String EXTRA_TEMP = "com.leafdigital.humidity.Temp";
	public static final String EXTRA_HUMIDITY_ERROR = "com.leafdigital.humidity.HumidityError";
	public static final String EXTRA_TEMP_ERROR = "com.leafdigital.humidity.TempError";
	public static final String EXTRA_SAVEBANK = "com.leafdigital.humidity.SaveBank";

	public final static String PREFS_BANK_NAME = "bankName";

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
     findViewById(R.id.reading).setOnClickListener(new OnClickListener()
    {
			@Override
			public void onClick(View v)
			{
				startActivityForResult(
					new Intent(HumidityActivity.this, TakeReadingActivity.class), 0);
			}
    });

    int[ ] ids = { R.id.stored1, R.id.stored2, R.id.stored3,
     R.id.stored4, R.id.stored5, R.id.stored6 };
    for(int i=0; i<6; i++)
    {
     final int storeNum = i;
     Button storedButton = (Button)findViewById(ids[i]);
			storedButton.setOnClickListener(new View.OnClickListener()
			{
				@Override
				public void onClick(View v)
				{
				Intent intent = new Intent(HumidityActivity.this, ShowRecordsActivity.class);
				intent.putExtra(EXTRA_SAVEBANK, storeNum);
				startActivity(intent);
				}
			});
    }
  }
```

```
@Override
protected void onResume()
{
   int[ ] ids = { R.id.stored1, R.id.stored2, R.id.stored3,
    R.id.stored4, R.id.stored5, R.id.stored6 };
   for(int i=0; i<6; i++)
   {
    Button storedButton = (Button)findViewById(ids[i]);
    storedButton.setText(HumidityActivity.getBankName(this, i));
   }
    super.onResume();
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    // Ignore anything but okay
          if(resultCode != RESULT_OK)
          {
                return;
          }

          if(data != null && data.getIntExtra(EXTRA_SAVEBANK, -1) != -1)
          {
                save(
                      data.getIntExtra(EXTRA_SAVEBANK, -1),
                      System.currentTimeMillis(),
                      data.getDoubleExtra(EXTRA_TEMP, 0.0),
                      data.getFloatExtra(EXTRA_TEMP_ERROR, 0.0f),
                      data.getDoubleExtra(EXTRA_HUMIDITY, 0.0),
                      data.getFloatExtra(EXTRA_HUMIDITY_ERROR, 0.0f));

                Intent intent = new Intent(HumidityActivity.this, ShowRecordsActivity.class);
          intent.putExtra(EXTRA_SAVEBANK, data.getIntExtra(EXTRA_SAVEBANK, -1));
          startActivity(intent);
          }
}

private void save(int bank, long time, double temp, float tempError,
    double humidity, float humidityError)
{
    SQLiteDatabase db = new RecordsOpenHelper(this).getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("bank", bank);
    values.put("javatime", time);
    values.put("temperature", temp);
    values.put("humidity", humidity);
    values.put("temperature_error", tempError);
    values.put("humidity_error", humidityError);
    db.insert("readings", null, values);
    db.close();
```

```
  }

  public static String getBankName(Activity activity, int bank)
  {
      return activity.getSharedPreferences("com.leafdigital.humidity", MODE_PRIVATE).getString(
           PREFS_BANK_NAME + bank, "" + (bank+1));
  }
}
```

系统主界面的执行效果如图 12-1 所示。

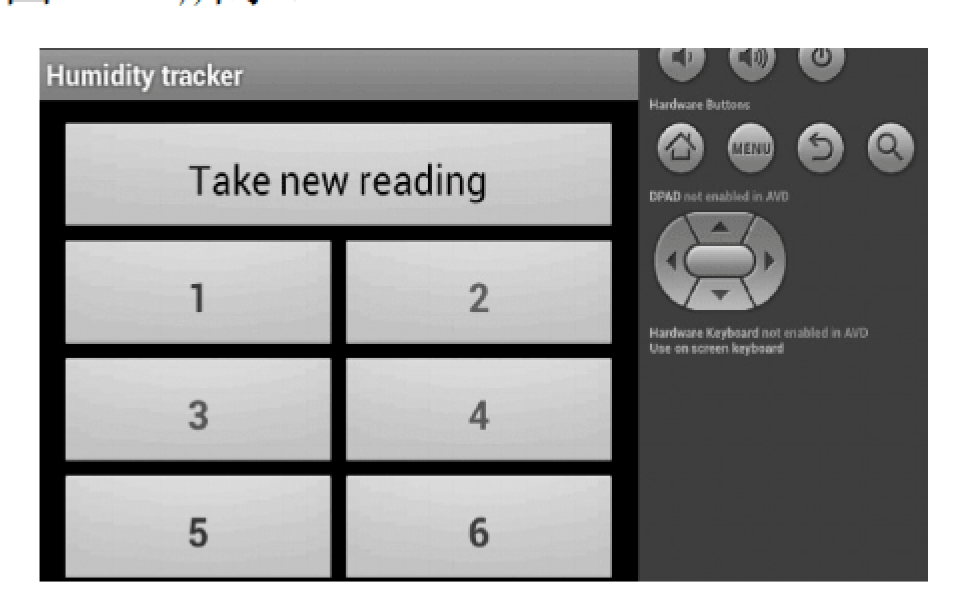


图 12-1　系统主界面的执行效果

12.2　设置具体值

知识点讲解：光盘:视频\知识点\第 12 章\设置具体值.avi

当单击 Take new reading 按钮后会弹出设置具体值界面，此界面的布局文件是 takereading.xml，在界面中可以设置温度和湿度值，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:padding="10dp"
     >

<TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
      android:text="@string/temp"
      style="@style/BigFont"
      />

<LinearLayout
     android:orientation="horizontal"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
      >

      <EditText
           android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:id="@+id/temp"
            android:inputType="numberDecimal"
            android:width="100dp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:text="@string/degreesc"
            style="@style/BigFont"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:text="@string/plusminus"
            style="@style/BigFont"
            />

        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/temperror"
            android:inputType="numberDecimal"
            android:width="60dp"
            />

</LinearLayout>

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/humidity"
    android:paddingTop="15dp"
    style="@style/BigFont"
    />

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:paddingBottom="30dp"
    >

        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/humidity"
```

```
            android:inputType="numberDecimal"
            android:width="100dp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:text="@string/percent"
            style="@style/BigFont"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:text="@string/plusminus"
            style="@style/BigFont"
            />

        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/humidityerror"
            android:inputType="numberDecimal"
            android:width="60dp"
            />

</LinearLayout>

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/ok"
    android:text="@string/ok"
    android:enabled="false"
    style="@style/BigFont"
    />

</LinearLayout>
```

对应的程序文件是 TakeReadingActivity.java，功能是获取用户设置的温度值和湿度值进行保存，具体实现代码如下所示。

```
public class TakeReadingActivity extends Activity
{
    private final static String DECIMAL_REGEX = "[0-9]+(\\.[0-9]+)?";

    private final static String PREFS_TEMP_ERROR = "tempError",
        PREFS_HUMIDITY_ERROR = "humidityError";

    private Button ok;
```

```
    private EditText tempEdit, humidityEdit, tempErrorEdit, humidityErrorEdit;

    /**
     * Rounds a value for display. This rounds to two decimal places, but then
     * discards the decimal places if not used, so that it can display results
     * of the form 0.97, 0.9, and 0.
     * @param value Value
     * @return String
     */
    private static String roundValue(float value)
    {
        String s = String.format("%.2f", value);
        while(s.endsWith("0"))
        {
            s = s.substring(0, s.length() - 1);
        }
        if(s.endsWith("."))
        {
            s = s.substring(0, s.length() - 1);
        }
        return s;
    }

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.takereading);

    ok = (Button)findViewById(R.id.ok);
    tempEdit = (EditText)findViewById(R.id.temp);
    tempErrorEdit = (EditText)findViewById(R.id.temperror);
    humidityEdit = (EditText)findViewById(R.id.humidity);
    humidityErrorEdit = (EditText)findViewById(R.id.humidityerror);

    // Set default error values
    SharedPreferences prefs = getSharedPreferences("com.leafdigital.humidity",MODE_PRIVATE);
    tempErrorEdit.setText(roundValue(prefs.getFloat(PREFS_TEMP_ERROR, 1.0f)));
    humidityErrorEdit.setText(roundValue(prefs.getFloat(PREFS_HUMIDITY_ERROR, 4.0f)));

    TextWatcher watcher = new TextWatcher()
    {
            @Override
            public void afterTextChanged(Editable arg0)
            {
                textChanged();
            }

            @Override
            public void beforeTextChanged(CharSequence arg0, int arg1, int arg2,
```

```
                int arg3)
            {
            }

            @Override
            public void onTextChanged(CharSequence arg0, int arg1, int arg2, int arg3)
            {
            }
    };
    tempEdit.addTextChangedListener(watcher);
    tempErrorEdit.addTextChangedListener(watcher);
    humidityEdit.addTextChangedListener(watcher);
    humidityErrorEdit.addTextChangedListener(watcher);

    ok.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
            // User has entered a reading
            double temp = Double.parseDouble(tempEdit.getText().toString());
            double humidity = Double.parseDouble(humidityEdit.getText().toString());

            float tempError = Float.parseFloat(tempErrorEdit.getText().toString());
            float humidityError = Float.parseFloat(humidityErrorEdit.getText().toString());

            // Update preferences to track the error
            SharedPreferences prefs = getSharedPreferences("com.leafdigital.humidity",MODE_PRIVATE);
            if(prefs.getFloat(PREFS_TEMP_ERROR, 1.0f) != tempError
                || prefs.getFloat(PREFS_HUMIDITY_ERROR, 1.0f) != humidityError)
            {
                SharedPreferences.Editor editor = prefs.edit();
                editor.putFloat(PREFS_TEMP_ERROR, tempError);
                editor.putFloat(PREFS_HUMIDITY_ERROR, humidityError);
                editor.commit();
            }

                Intent intent = new Intent(TakeReadingActivity.this, ShowReadingActivity.class);
                intent.putExtra(EXTRA_TEMP, temp);
                intent.putExtra(EXTRA_TEMP_ERROR, tempError);
                intent.putExtra(EXTRA_HUMIDITY, humidity);
                intent.putExtra(EXTRA_HUMIDITY_ERROR, humidityError);

                startActivityForResult(intent, 0);
            }
        });
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
```

```
        // Pass result data through, and finish
                if(resultCode == RESULT_OK)
                {
                setResult(resultCode, data);
                finish();
                }
    }

    private void textChanged()
    {
        String tempString = tempEdit.getText().toString();
                String tempErrorString = tempErrorEdit.getText().toString();
                String humidityString = humidityEdit.getText().toString();
                String humidityErrorString = humidityErrorEdit.getText().toString();
                ok.setEnabled(
                tempString.matches(DECIMAL_REGEX)
                && tempErrorString.matches(DECIMAL_REGEX)
                && humidityString.matches(DECIMAL_REGEX)
                && humidityErrorString.matches(DECIMAL_REGEX)
                && Double.parseDouble(tempString) <= 40.0
                && Double.parseDouble(tempErrorString) <= 10.0
                && Double.parseDouble(humidityString) <= 100.0
                && Double.parseDouble(humidityString) >= 1.0
                && Double.parseDouble(humidityErrorString) <= 10.0);
    }
}
```

系统设置界面的执行效果如图 12-2 所示。

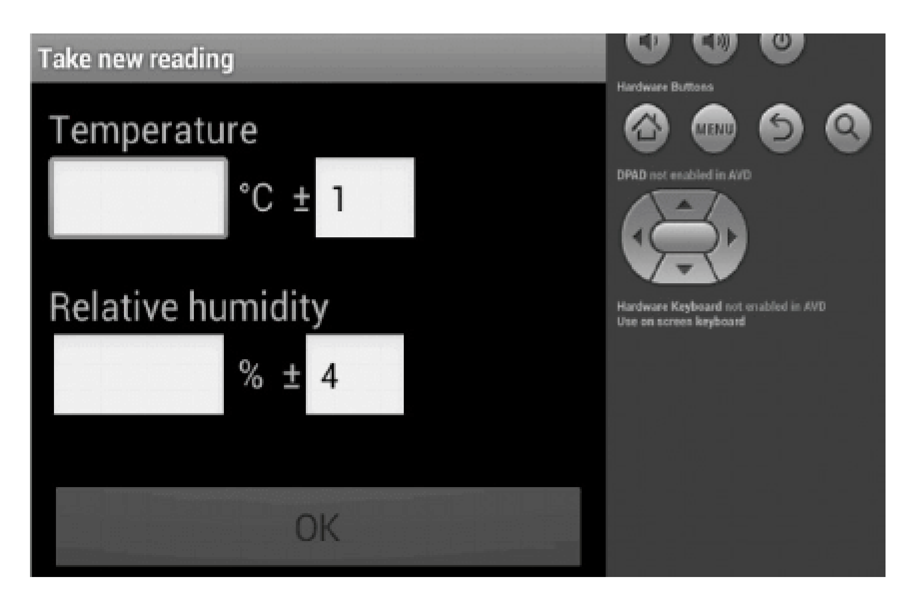


图 12-2　系统设置界面的执行效果

12.3　显示当前的值

知识点讲解：光盘:视频\知识点\第 12 章\显示当前的值.avi

布局文件 showreading.xml 的功能是，通过文本控件显示当前的温度、湿度、露点（大气中的湿空气由于温度下降，使所含的水蒸气达到饱和状态而开始凝结时的温度）和水密度值。文件 showreading.xml 的具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
```



```
    android:layout_height="fill_parent"
    android:padding="10dp"
    >

<LinearLayout
    android:orientation="horizontal"
    style="@style/ShowTableRow"
     >
     <TextView
          android:text="@string/temp"
          style="@style/ShowTableRowLabel"
          />
     <TextView
          android:id="@+id/showTempL"
          style="@style/ShowTableRowValue"
          />
     <TextView
          style="@style/ShowTableRowTo"
          />
     <TextView
          android:id="@+id/showTempH"
          style="@style/ShowTableRowValue2"
          />
     <TextView
          android:text="@string/degreesc"
          style="@style/ShowTableRowUnit"
          />
</LinearLayout>

<LinearLayout
    android:orientation="horizontal"
    style="@style/ShowTableRow"
     >
     <TextView
          android:text="@string/humidity"
          style="@style/ShowTableRowLabel"
          />
     <TextView
          android:id="@+id/showHumidityL"
          style="@style/ShowTableRowValue"
          />
     <TextView
          style="@style/ShowTableRowTo"
          />
     <TextView
          android:id="@+id/showHumidityH"
          style="@style/ShowTableRowValue2"
          />
     <TextView
          android:text="@string/percent"
          style="@style/ShowTableRowUnit"
```

```
            />
</LinearLayout>

<LinearLayout
     android:orientation="horizontal"
     style="@style/ShowTableRow"
      >
      <TextView
            android:text="@string/dewpoint"
            style="@style/ShowTableRowLabel"
            />
      <TextView
            android:id="@+id/showDewpointL"
            style="@style/ShowTableRowValue"
            />
      <TextView
            style="@style/ShowTableRowTo"
            />
      <TextView
            android:id="@+id/showDewpointH"
            style="@style/ShowTableRowValue2"
            />
      <TextView
            android:text="@string/degreesc"
            style="@style/ShowTableRowUnit"
            />
</LinearLayout>

<LinearLayout
     android:orientation="horizontal"
     style="@style/ShowTableRow"
      >
      <TextView
            android:text="@string/density"
            style="@style/ShowTableRowLabel"
            />
      <TextView
            android:id="@+id/showDensityL"
            style="@style/ShowTableRowValue"
            />
      <TextView
            style="@style/ShowTableRowTo"
            />
      <TextView
            android:id="@+id/showDensityH"
            style="@style/ShowTableRowValue2"
            />
      <TextView
            android:text="@string/gm3"
            style="@style/ShowTableRowUnit"
            />
```

```
</LinearLayout>

<LinearLayout
    android:paddingTop="15dp"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ok"
        android:text="@string/ok"
        style="@style/BigFont"
        android:layout_weight="1"
        />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/save"
        android:text="@string/save"
        style="@style/BigFont"
        android:layout_weight="1"
        />
</LinearLayout>
</LinearLayout>
```

程序文件 ShowReadingActivity.java 的功能是获取并显示当前温度、湿度、露点和水密度的值，具体实现代码如下所示。

```
public class ShowReadingActivity extends Activity
{

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.showreading);

    double temp = getIntent().getDoubleExtra(EXTRA_TEMP, 0.0);
    float tempError = getIntent().getFloatExtra(EXTRA_TEMP_ERROR, 0.0f);
    double humidity = getIntent().getDoubleExtra(EXTRA_HUMIDITY, 0.0);
    float humidityError = getIntent().getFloatExtra(EXTRA_HUMIDITY_ERROR, 0.0f);

    ClimateData data = new ClimateData(temp, tempError, humidity, humidityError);

        FieldValue value = data.getField(FIELD_TEMPERATURE);
    ((TextView)findViewById(R.id.showTempL)).setText(
     String.format("%.1f", value.getMin()));
    ((TextView)findViewById(R.id.showTempH)).setText(
     String.format("%.1f", value.getMax()));
```

```
        value = data.getField(FIELD_HUMIDITY);
((TextView)findViewById(R.id.showHumidityL)).setText(
 String.format("%.1f", value.getMin()));
((TextView)findViewById(R.id.showHumidityH)).setText(
 String.format("%.1f", value.getMax()));

        value = data.getField(FIELD_DEWPOINT);
((TextView)findViewById(R.id.showDewpointL)).setText(
 String.format("%.1f", value.getMin()));
((TextView)findViewById(R.id.showDewpointH)).setText(
 String.format("%.1f", value.getMax()));

        value = data.getField(FIELD_DENSITY);
((TextView)findViewById(R.id.showDensityL)).setText(
 String.format("%.2f", value.getMin()));
((TextView)findViewById(R.id.showDensityH)).setText(
 String.format("%.2f", value.getMax()));

findViewById(R.id.ok).setOnClickListener(new OnClickListener()
        {
                @Override
                public void onClick(View v)
                {
                Intent intent = new Intent();
                intent.putExtra(EXTRA_SAVEBANK, -1);
                        setResult(RESULT_OK, intent);
                        finish();
                }
        });

findViewById(R.id.save).setOnClickListener(new OnClickListener()
        {
                @Override
                public void onClick(View v)
                {
                // User has selected to save the reading
                Intent intent = new Intent(ShowReadingActivity.this, SaveReadingActivity.class);
                intent.putExtra(EXTRA_TEMP, getIntent().getDoubleExtra(EXTRA_TEMP, 0.0));
                intent.putExtra(EXTRA_TEMP_ERROR, getIntent().getFloatExtra(EXTRA_TEMP_ERROR, 0.0f));
                intent.putExtra(EXTRA_HUMIDITY, getIntent().getDoubleExtra(EXTRA_HUMIDITY, 0.0));
                intent.putExtra(EXTRA_HUMIDITY_ERROR, getIntent().getFloatExtra(EXTRA_HUMIDITY_ERROR,
                0.0f));
                startActivityForResult(intent, 0);
                }
        });
 }

 @Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data)
 {
 // Pass result data through, and finish
```

```
            if(resultCode == RESULT_OK)
            {
            setResult(resultCode, data);
            finish();
            }
        }
}
```

系统显示当前值界面的执行效果如图 12-3 所示。

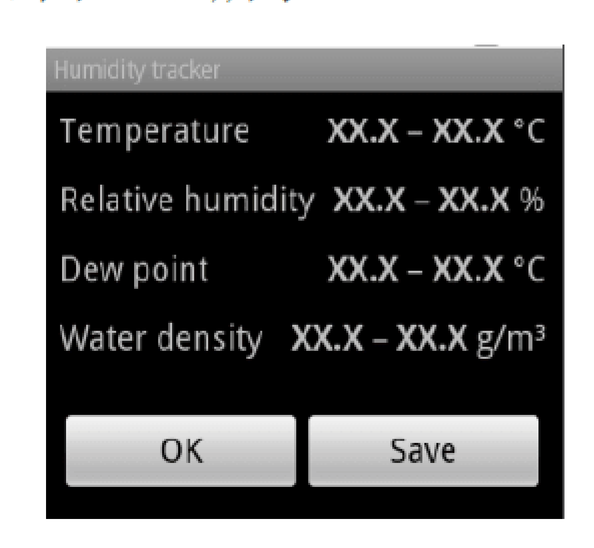


图 12-3　显示当前值界面的执行效果

12.4　保存当前数值

知识点讲解：光盘:视频\知识点\第 12 章\保存当前数值.avi

如果在当前值界面中单击 Save 按钮，系统会保存当前的测试数值。本节将详细讲解本系统保存当前数值模块的具体实现流程。

12.4.1　实现布局文件

布局文件 savereading.xml 的功能是保存系统当前测试的湿度和温度值，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    >
<TableLayout
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:layout_weight="1"
     android:stretchColumns="0,1"
     >
     <TableRow>
          <Button
               android:id="@+id/stored1"
               android:textColor="@color/stored1"
               style="@style/StoredButton"
               />
          <Button
```

```
                android:id="@+id/stored2"
                android:textColor="@color/stored2"
                style="@style/StoredButton"
                />
    </TableRow>
    <TableRow>
        <Button
                android:id="@+id/stored3"
                android:textColor="@color/stored3"
                style="@style/StoredButton"
                />
        <Button
                android:id="@+id/stored4"
                android:textColor="@color/stored4"
                style="@style/StoredButton"
                />
    </TableRow>
    <TableRow>
        <Button
                android:id="@+id/stored5"
                android:textColor="@color/stored5"
                style="@style/StoredButton"
                />
        <Button
                android:id="@+id/stored6"
                android:textColor="@color/stored6"
                style="@style/StoredButton"
                />
    </TableRow>
</TableLayout>
</LinearLayout>
```

12.4.2　实现 SaveReadingActivity

程序文件 SaveReadingActivity.java 的功能是保存当前的值到数据库中，具体实现代码如下所示。

```
public class SaveReadingActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
   setContentView(R.layout.savereading);

   int[ ] ids = { R.id.stored1, R.id.stored2, R.id.stored3,
    R.id.stored4, R.id.stored5, R.id.stored6 };
   for(int i=0; i<6; i++)
   {
    final int storeNum = i;
    Button storedButton = (Button)findViewById(ids[i]);
```

```
        storedButton.setText(HumidityActivity.getBankName(this, i));
                storedButton.setOnClickListener(new View.OnClickListener()
                {
                    @Override
                    public void onClick(View v)
                    {
                    Intent intent = new Intent();
                    intent.putExtra(EXTRA_TEMP, getIntent().getDoubleExtra(EXTRA_TEMP, 0.0));
                    intent.putExtra(EXTRA_TEMP_ERROR, getIntent().getFloatExtra(EXTRA_TEMP_
ERROR, 0.0f));
                    intent.putExtra(EXTRA_HUMIDITY, getIntent().getDoubleExtra(EXTRA_HUMIDITY, 0.0));
                    intent.putExtra(EXTRA_HUMIDITY_ERROR, getIntent().getFloatExtra(EXTRA_
HUMIDITY_ERROR, 0.0f));
                    intent.putExtra(EXTRA_SAVEBANK, storeNum);
                        setResult(RESULT_OK, intent);
                        finish();
                    }
                });
        }
        }

}
```

12.5　图形化显示测试结果

知识点讲解：光盘:视频\知识点\第 12 章\图形化显示测试结果.avi

布局文件 showrecords.xml 的功能是以图形化的方式显示当前的测试结果，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/layout"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:padding="0dp"
      >
      <!-- Graph component added programatically -->
</LinearLayout>
```

程序文件 ShowRecordsActivity.java 的功能是监听用户的操作事件，根据用户的选项绘制图形化界面，以图形化方式显示当前的检测结果值。文件 ShowRecordsActivity.java 的具体实现代码如下所示。

```
public class ShowRecordsActivity extends Activity
{
        private int bank;
        private DataPoint[ ] data;
        private RecordsGraph graph;
        private int graphField;

        private int nearestPointIndex;
```

```
private final static String PREFS_CURRENT_GRAPH = "currentGraph";

public final static float[ ] MAX_RANGE = { 40.0f, 100.0f, 40.0f, 50.0f };
public final static float[ ] RANGE_STEP = { 5.0f, 5.0f, 5.0f, 5.0f };
public final static int[ ] FIELD_LABEL = { R.string.temp, R.string.humidity,
        R.string.dewpoint, R.string.density };
public final static int[ ] FIELD_UNIT = { R.string.degreesc, R.string.percent,
        R.string.degreesc, R.string.gm3 };

private final static float HIT_RADIUS = 48;
private final static float GRAPH_PADDING = 10;

private final static long ONE_DAY = 24L * 3600L * 1000L;

private final static int DIALOG_POINT=1, DIALOG_NAME=2, DIALOG_DELETE=3;

private static class DataPoint
{
        long javaTime;
        ClimateData data;

        private DataPoint(long javaTime, ClimateData data)
        {
                this.javaTime = javaTime;
                this.data = data;
        }
}

private class RecordsGraph extends View
{
        private float xScale, yScale, left, bottom, density;
        private long startTime;

        public RecordsGraph()
        {
                super(ShowRecordsActivity.this);
                setLayoutParams(new LinearLayout.LayoutParams(
                        LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT, 1.0f));
        }

        @Override
        public boolean onTouchEvent(MotionEvent event)
        {
                switch(event.getAction())
                {
                case MotionEvent.ACTION_DOWN:
                case MotionEvent.ACTION_MOVE:
                        updateNearestPoint(event.getX(), event.getY());
                        return true;
                case MotionEvent.ACTION_UP:
```

```
            updateNearestPoint(event.getX(), event.getY());
            if(nearestPointIndex != -1)
            {
                showDialog(DIALOG_POINT);
            }
            nearestPointIndex = -1;
            invalidate();
            return true;
        }
        return false;
    }

    private void updateNearestPoint(float x, float y)
    {
        // Crap algorithm, but let's just try all the points within any kind of
        // range.
        float radius = HIT_RADIUS * density;
        float bestDiff = radius * radius;
        int bestIndex = -1;
        for(int i=0; i<data.length; i++)
        {
            // Get X point and check if within range
            float pointX = (float)(data[i].javaTime - startTime) * xScale + left;
            if (pointX > x + radius)
            {
                break;
            }
            if (pointX < x - radius)
            {
                continue;
            }

            // Find Y point
            FieldValue field = data[i].data.getField(graphField);
            float pointY = bottom - field.getMeasured() * yScale;

            // Work out difference (= distance squared)
            float diff = (pointY - y) * (pointY - y) + (pointX - x) * (pointX - x);
            if (diff < bestDiff)
            {
                bestDiff = diff;
                bestIndex = i;
            }
        }

        if(bestIndex != nearestPointIndex)
        {
            nearestPointIndex = bestIndex;
            invalidate();
        }
```

```
    }

    @Override
    protected void onDraw(Canvas canvas)
    {
        DisplayMetrics metrics = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(metrics);
        density = metrics.density;
        float blobRadius = 2f * density;
        float padding = GRAPH_PADDING * density;

        // Clear to off-black
        canvas.drawColor(Color.argb(255, 30, 30, 30));

        // Get available width and height for graph
        float w = getWidth() - padding*2, h = getHeight() - padding*2;
        left = padding;
        bottom = getHeight() - padding;

        // Draw axes in grey
        Paint grey = new Paint(Paint.ANTI_ALIAS_FLAG);
        grey.setColor(Color.argb(255, 128, 128, 128));
        canvas.drawLine(left, bottom, left+w, bottom, grey);
        canvas.drawLine(left, bottom, left, bottom-h, grey);
        canvas.drawLine(left, bottom-h, left+ 1.5f*blobRadius, bottom-h, grey);

        // Work out Y scaling
        float actualMax = 0;
        for(int i=0; i<data.length; i++)
        {
            float value = data[i].data.getField(graphField).getMax();
            if(value > actualMax)
            {
                actualMax = value;
            }
        }
        float range = MAX_RANGE[graphField];
        if(data.length == 0)
        {
            actualMax = range;
        }
        while(range - RANGE_STEP[graphField] > actualMax)
        {
            range -= RANGE_STEP[graphField];
        }

        yScale = h / range;

        // Draw scale text
        String max = String.format("%.0f", range);
```

```
            grey.setTextSize(12f * density);
            Rect bounds = new Rect();
            grey.getTextBounds(max, 0, max.length(), bounds);
            canvas.drawText(max, left + blobRadius * 2 * density,
                bottom - h - bounds.top, grey);

            // If there is no data, stop here
            if(data.length == 0)
            {
                return;
            }

            // Work out X scaling
            startTime = data[0].javaTime;
            long timeRange = data[data.length - 1].javaTime - startTime;
            if(timeRange == 0)
            {
                timeRange = 1;
            }
            xScale = w / (float)timeRange;

            // Draw each point
            Paint white = new Paint(Paint.ANTI_ALIAS_FLAG);
            white.setColor(Color.argb(255, 255, 255, 255));
            Paint faintLine = new Paint(Paint.ANTI_ALIAS_FLAG);
            faintLine.setColor(Color.argb(160, 0, 255, 0));
            for(int i=0; i<data.length; i++)
            {
                float x = (float)(data[i].javaTime - startTime) * xScale;
                FieldValue field = data[i].data.getField(graphField);
                float yMin = field.getMin() * yScale, yMax = field.getMax() * yScale;
                float y = field.getMeasured() * yScale;
                if (i==nearestPointIndex)
                {
                    Paint faint = new Paint(Paint.ANTI_ALIAS_FLAG);
                    faint.setColor(Color.argb(64,255,255,255));
                    canvas.drawCircle(x + left, bottom - y, blobRadius*20, faint);
                }

                canvas.drawLine(x + left, bottom - yMin, x + left, bottom - yMax, faintLine);
                canvas.drawCircle(x + left, bottom - y, blobRadius, white);
            }
        }

        @Override
        protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)
        {
            int w;
            if(MeasureSpec.getMode(widthMeasureSpec) == MeasureSpec.UNSPECIFIED)
            {
```

```
                w = 100;
            }
            else
            {
                w = MeasureSpec.getSize(widthMeasureSpec);
            }

            int h;
            if(MeasureSpec.getMode(heightMeasureSpec) == MeasureSpec.UNSPECIFIED)
            {
                h = 100;
            }
            else
            {
                h = MeasureSpec.getSize(heightMeasureSpec);
            }
            setMeasuredDimension(w, h);
        }
    }

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        // Get current bank and set title
        bank = getIntent().getIntExtra(HumidityActivity.EXTRA_SAVEBANK, -1);

        loadData();

    setContentView(R.layout.showrecords);
    LinearLayout layout = (LinearLayout)findViewById(R.id.layout);

    graph = new RecordsGraph();
    layout.addView(graph, 0);

    // Get default field
    setField(getSharedPreferences("com.leafdigital.humidity", MODE_PRIVATE).getInt(
     PREFS_CURRENT_GRAPH, FIELD_HUMIDITY));
    }

    private void loadData()
    {
        // Load records. (Note: This uses a writable database because it might need
        // to do a db version update, so it has to be able to write)
        SQLiteDatabase db = new RecordsOpenHelper(this).getWritableDatabase();
        Cursor cursor = db.query(
            "readings", new String[ ] {  "javatime", "temperature", "temperature_error",
                "humidity", "humidity_error" },
```

```
            "bank = ?",    new String[ ] { bank + "" },    null, null, "javatime");
        int count = cursor.getCount();
        data = new DataPoint[count];
        for(int i=0; i<count; i++)
        {
            cursor.moveToNext();
            data[i] = new DataPoint(cursor.getLong(0), new ClimateData(cursor.getFloat(1),
                cursor.getFloat(2),cursor.getFloat(3),cursor.getFloat(4)));
        }
        cursor.close();
        db.close();

        // Clear other fields that depend on data
        nearestPointIndex = -1;
    }

    private void setField(int field)
    {
        this.graphField = field;
        setTitle(HumidityActivity.getBankName(this, bank) + " - " + getString(FIELD_LABEL[field])
            + " (" + getString(FIELD_UNIT[field]) + ")");
        graph.invalidate();

        SharedPreferences prefs = getSharedPreferences("com.leafdigital.humidity", MODE_PRIVATE);
        if(prefs.getInt(PREFS_CURRENT_GRAPH, FIELD_HUMIDITY) != field)
        {
            if(field == FIELD_HUMIDITY)
            {
                prefs.edit().remove(PREFS_CURRENT_GRAPH).commit();
            }
            else
            {
                prefs.edit().putInt(PREFS_CURRENT_GRAPH, field).commit();
            }
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.showrecords, menu);
        return true;
    }

    @Override
    public boolean onPrepareOptionsMenu(Menu menu)
    {
        int id = -1;
        switch(graphField)
```

```
        {
        case FIELD_TEMPERATURE :
            id = R.id.temp;
            break;
        case FIELD_HUMIDITY :
            id = R.id.humidity;
            break;
        case FIELD_DEWPOINT :
            id = R.id.dewpoint;
            break;
        case FIELD_DENSITY :
            id = R.id.density;
            break;
        }
        menu.findItem(id).setChecked(true);

        menu.findItem(R.id.delete).setEnabled(data.length != 0);
        long now = System.currentTimeMillis();
        menu.findItem(R.id.deletebefore1d).setEnabled(data.length>0
            && now - data[0].javaTime > 1 * ONE_DAY);
        menu.findItem(R.id.deletebefore7d).setEnabled(data.length>0
            && now - data[0].javaTime > 7 * ONE_DAY);
        menu.findItem(R.id.deletebefore30d).setEnabled(data.length>0
            && now - data[0].javaTime > 30 * ONE_DAY);

        return super.onPrepareOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        switch(item.getItemId())
        {
        case R.id.temp :
            setField(FIELD_TEMPERATURE);
            return true;
        case R.id.humidity :
            setField(FIELD_HUMIDITY);
            return true;
        case R.id.dewpoint :
            setField(FIELD_DEWPOINT);
            return true;
        case R.id.density :
            setField(FIELD_DENSITY);
            return true;
        case R.id.name :
            showDialog(DIALOG_NAME);
            return true;
        case R.id.deletelast :
            deleteData(data[data.length-1].javaTime-1, getString(R.string.deletelast), true);
```

```
                return true;
            case R.id.deletebefore1d :
                deleteData(System.currentTimeMillis() - ONE_DAY, getString(R.string.deletebefore1d), false);
                return true;
            case R.id.deletebefore7d :
                deleteData(System.currentTimeMillis() - 7 * ONE_DAY, getString(R.string.deletebefore7d), false);
                return true;
            case R.id.deletebefore30d :
                deleteData(System.currentTimeMillis() - 30 * ONE_DAY, getString(R.string.deletebefore30d), false);
                return true;
            case R.id.deleteall :
                deleteData(System.currentTimeMillis() + 1, getString(R.string.deleteall), false);
                return true;
            default:
                return super.onOptionsItemSelected(item);
            }
        }

        @Override
        protected Dialog onCreateDialog(int id)
        {
            // Create dialog
            final Dialog dialog = new Dialog(this);
            switch(id)
            {
            case DIALOG_POINT :
                dialog.setContentView(R.layout.pointdialog);
                View ok = dialog.findViewById(R.id.ok);
                ok.setOnClickListener(new OnClickListener()
                {
                    @Override
                    public void onClick(View v)
                    {
                        dialog.cancel();
                    }
                });
                break;

            case DIALOG_NAME :
                dialog.setContentView(R.layout.namedialog);
                dialog.setTitle(R.string.name);
                final EditText edit = (EditText)dialog.findViewById(R.id.name);
                final View okName = dialog.findViewById(R.id.ok);
                okName.setOnClickListener(new OnClickListener()
                {
                    @Override
                    public void onClick(View v)
                    {
                        String name = edit.getText().toString();
                        SharedPreferences.Editor editor = getSharedPreferences("com.leafdigital.humidity",
```

```
MODE_PRIVATE).edit();
                    editor.putString(PREFS_BANK_NAME + bank, name);
                    editor.commit();
                    dialog.dismiss();
                    setField(graphField);
                }
            });
            edit.addTextChangedListener(new TextWatcher()
            {
                @Override
                public void onTextChanged(CharSequence s, int start, int before, int count)
                {
                }

                @Override
                public void beforeTextChanged(CharSequence s, int start, int count, int after)
                {
                }

                @Override
                public void afterTextChanged(Editable s)
                {
                    int length = edit.getText().length();
                    okName.setEnabled(length <= 6 && length >= 1);
                }
            });
            break;

        case DIALOG_DELETE:
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setMessage(getString(R.string.confirmdelete));
            builder.setPositiveButton(getString(R.string.delete),
                new DialogInterface.OnClickListener()
                {
                    @Override
                    public void onClick(DialogInterface dialog, int which)
                    {
                        confirmDelete();
                        dialog.dismiss();
                    }
                });
            builder.setNegativeButton(getString(R.string.cancel),
                new DialogInterface.OnClickListener()
            {
                @Override
                public void onClick(DialogInterface dialog, int which)
                {
                    dialog.cancel();
                }
            });
```

```
            builder.setTitle("Placeholder");
            return builder.create();
        }
        return dialog;
    }

    @Override
    protected void onPrepareDialog(int id, Dialog dialog)
    {
        switch(id)
        {
        case DIALOG_POINT :
            preparePointDialog(dialog);
            break;
        case DIALOG_NAME :
            prepareNameDialog(dialog);
            break;
        case DIALOG_DELETE :
            prepareDeleteDialog(dialog);
            break;
        }
    }

    private void preparePointDialog(Dialog dialog)
    {
        // Get data
        DataPoint point = data[nearestPointIndex];
        FieldValue field = point.data.getField(graphField);

        // Set title to date
        SimpleDateFormat date = new SimpleDateFormat("yyyy-MM-dd HH:mm");
        dialog.setTitle(date.format(new Date(point.javaTime)));

        // Set icon
        ImageView image = (ImageView)dialog.findViewById(R.id.icon);
        int icon = -1;
        switch(graphField)
        {
        case FIELD_TEMPERATURE:
            icon = R.drawable.menu_temp;
            break;
        case FIELD_HUMIDITY:
            icon = R.drawable.menu_humidity;
            break;
        case FIELD_DEWPOINT:
            icon = R.drawable.menu_dewpoint;
            break;
        case FIELD_DENSITY:
            icon = R.drawable.menu_density;
            break;
```

```
        }
        image.setImageResource(icon);

        // Set units
        for(int unitId : new int[ ] { R.id.unit1, R.id.unit2, R.id.unit3 })
        {
            ((TextView)dialog.findViewById(unitId)).setText(getString(FIELD_UNIT[graphField]));
        }

        // Set actual values
        String format = graphField == FIELD_DENSITY ? "%.2f" : "%.1f";
        ((TextView)dialog.findViewById(R.id.minValue)).setText(
            String.format(format, field.getMin()));
        ((TextView)dialog.findViewById(R.id.estValue)).setText(
            String.format(format, field.getMeasured()));
        ((TextView)dialog.findViewById(R.id.maxValue)).setText(
            String.format(format, field.getMax()));
    }

    private void prepareNameDialog(Dialog dialog)
    {
        ((EditText)dialog.findViewById(R.id.name)).setText(
            HumidityActivity.getBankName(this, bank));
        ((Button)dialog.findViewById(R.id.ok)).setEnabled(true);
    }

    private void prepareDeleteDialog(Dialog dialog)
    {
        dialog.setTitle(deleteMessage);
    }
    private long deleteTime;
    private boolean deleteAfter;
    private String deleteMessage;
    private void deleteData(long time, String message, boolean after)
    {
        // This is crappy, but I have to call showDislog and then use this stuff
        // later.
        this.deleteTime = time;
        this.deleteMessage = message;
        this.deleteAfter = after;
        showDialog(DIALOG_DELETE);
    }
    private void confirmDelete()
    {
        SQLiteDatabase db = new RecordsOpenHelper(this).getWritableDatabase();
        db.delete("readings", "javatime " + (deleteAfter ? ">" : "<")
            + " ? AND bank = ?", new String[ ] { deleteTime + "", bank + ""});
        db.close();
        loadData();
        graph.invalidate();
```

```
    }
}
```

当用户按下设备中的 MENU 后，会在屏幕下方弹出 6 个操作选项，系统会以图形化界面效果显示检测结果，执行后效果如图 12-4 所示。

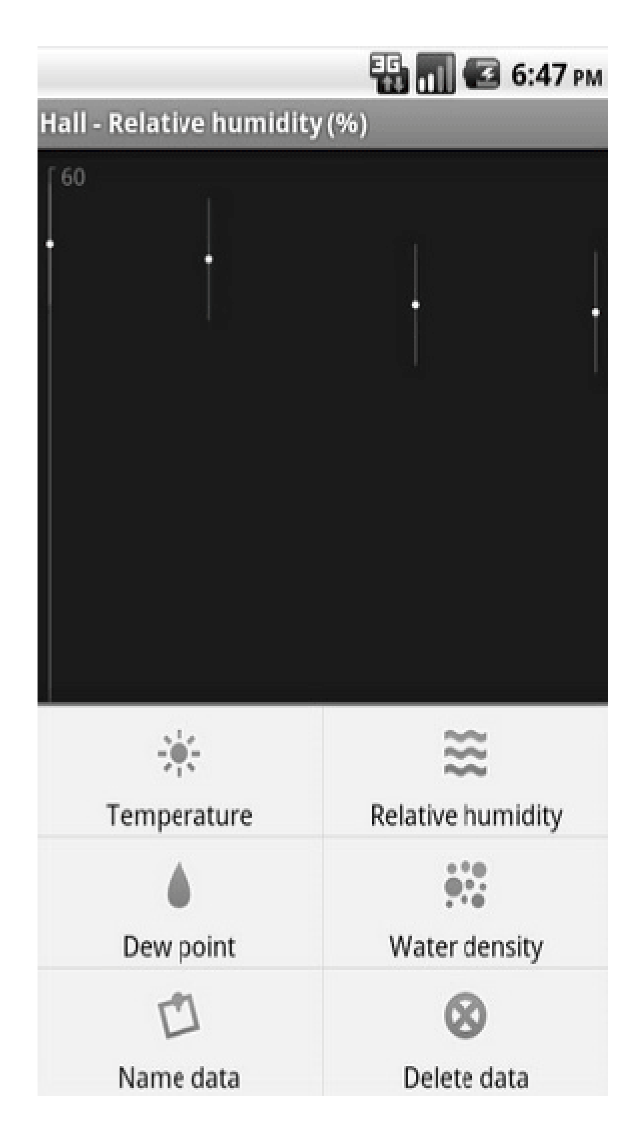


图 12-4　执行效果

12.6　湿度跟踪器

知识点讲解：光盘:视频\知识点\第 12 章\湿度跟踪器.avi

本系统检测湿度功能是通过开源代码湿度跟踪器实现的，开源文件 ClimateData.java 的具体实现代码如下所示。

```
public class ClimateData
{
    public final static int FIELD_TEMPERATURE=0, FIELD_HUMIDITY=1,
        FIELD_DEWPOINT=2, FIELD_DENSITY=3;

    private double[ ] measured = new double[4], min = new double[4], max = new double[4];

    public static class FieldValue
    {
        private float measured, min, max;

        private FieldValue(int field, double[ ] measured, double[ ] min, double[ ] max)
        {
            this.measured = (float)measured[field];
            this.min = (float)min[field];
            this.max = (float)max[field];
        }

        public float getMeasured()
```

```
            {
                return measured;
            }

            public float getMin()
            {
                return min;
            }

            public float getMax()
            {
                return max;
            }
        }

        public ClimateData(double temp, float tempError, double humidity, float humidityError)
        {
            measured[FIELD_TEMPERATURE] = temp;
            min[FIELD_TEMPERATURE] = Math.max(0, temp - tempError);
            max[FIELD_TEMPERATURE] = Math.min(40, temp + tempError);

            measured[FIELD_HUMIDITY] = humidity;
            min[FIELD_HUMIDITY] = Math.max(1, humidity - humidityError);
            max[FIELD_HUMIDITY] = Math.min(100, humidity + humidityError);

            measured[FIELD_DEWPOINT] = getDewpoint(measured);
            min[FIELD_DEWPOINT] = getDewpoint(min);
            max[FIELD_DEWPOINT] = getDewpoint(max);

            measured[FIELD_DENSITY] = getDensity(measured);
            min[FIELD_DENSITY] = getDensity(min);
            max[FIELD_DENSITY] = getDensity(max);
        }

    private static double getDewpoint(double[ ] values)
    {
        double temp = values[FIELD_TEMPERATURE];
        double humidity = values[FIELD_HUMIDITY];

            // Dewpoint valid up to 60 degrees, from
            // http://en.wikipedia.org/wiki/Dew_point#Calculating_the_dew_point
            double a = 17.271, b = 237.2;
            double gamma = ((a * temp) / (b + temp)) + Math.log(humidity/100.0);
            return b * gamma / (a - gamma);
    }

    private static double getDensity(double[ ] values)
    {
        double temp = values[FIELD_TEMPERATURE];
        double humidity = values[FIELD_HUMIDITY];
```

```
        // Approximate saturated vapour density valid up to 40 degrees, from
            // http://hyperphysics.phy-astr.gsu.edu/hbase/kinetic/relhum.html#c3
            double saturated = 5.018 + 0.32321 * temp + 0.0081847 * temp * temp
                + 0.00031243 * temp * temp * temp;
            return saturated * (humidity / 100.0);
    }

    public FieldValue getField(int field)
    {
        return new FieldValue(field, measured, min, max);
    }
}
```

到此为止，本实例的主要功能模块介绍完毕。和数据库操作相关的代码请读者参阅本书附带光盘中的具体源码，为节省篇幅，在此不再进行详细讲解。

第 13 章　小米录音机

在过去的几年中，小米手机创造了国产 Android 手机的销售神话。在小米手机中内置了一个 UI 界面优美、功能强大的录音机程序：MIUI 录音机。在 2012 年，MIUI 录音机程序完全开源，目的是集思广益，吸引广大程序员提高其功能和效果。本章将详细讲解 MIUI 录音机程序的基本源码，为读者学习本书后面的知识打下基础。

13.1　系 统 介 绍

知识点讲解：光盘:视频\知识点\第 13 章\系统介绍.avi

MIUI 录音机是一款基于 Android 原生录音机开发的产品，与 Android 系统自带的原生录音机相比，MIUI 录音机在 UI、交互方式和录音效果上都有较大的提升，其磁带的显示效果更是深受广大用户的好评。MIUI 录音机由 MIUI 团队（www.miui.com）发起并贡献第一批代码，遵循 NOTICE 文件所描述的开源协议，开源后为 MiCode 社区（www.micode.net）所拥有，并由社区发布和维护。

Bug 反馈和跟踪，可访问如下所示的 GitHub 地址，如图 13-1 所示。

```
https://github.com/MiCode/SoundRecorder/issues?sort=created&direction=desc&state=open
```

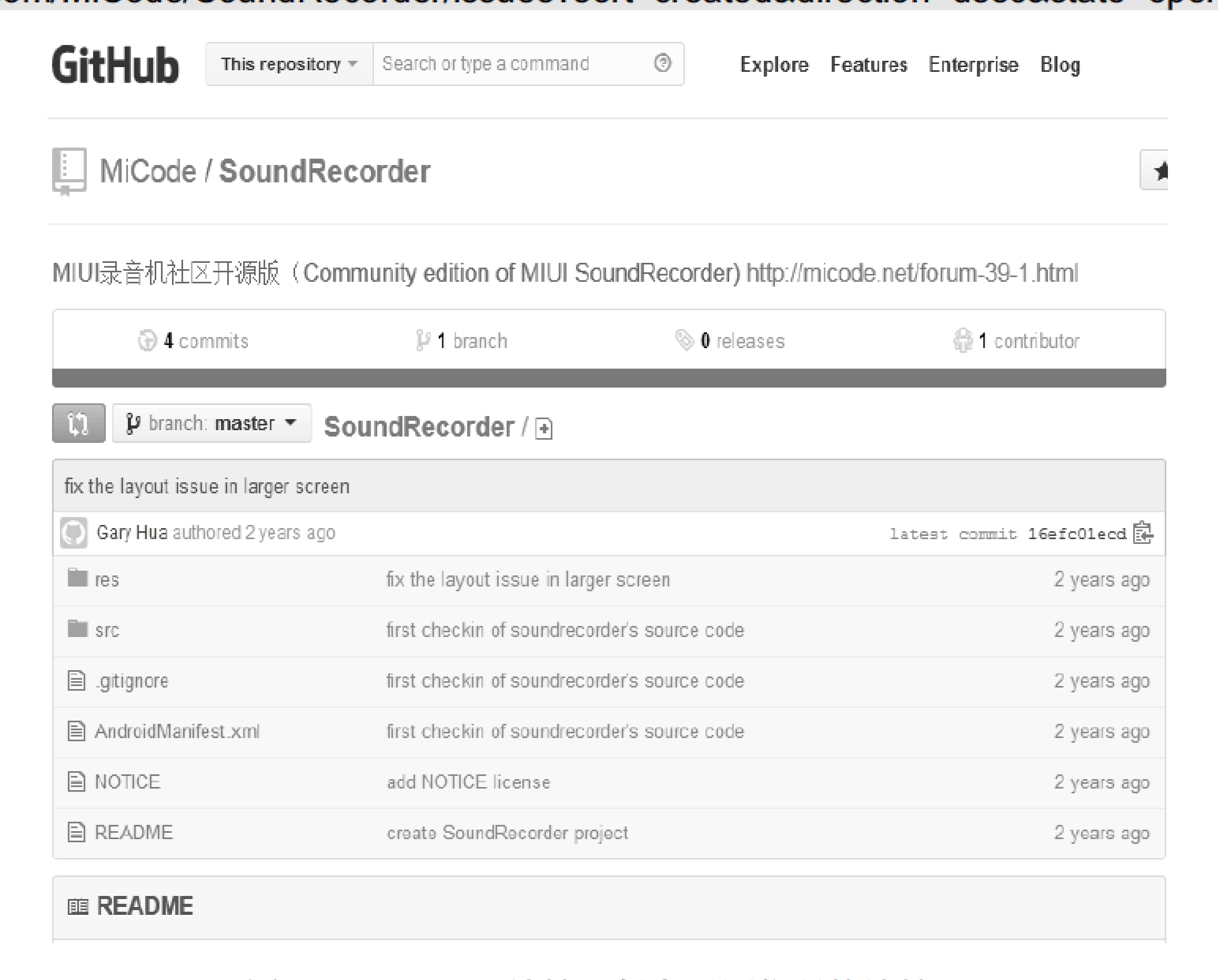


图 13-1　GitHub 地址（保存开源代码的地址）

功能建议和综合讨论，可访问如下所示的 MiCode 地址，如图 13-2 所示。

```
http://micode.net/forum-39-1.html
```



图 13-2　用户和开发者讨论区

13.2　系统主界面

知识点讲解：光盘:视频\知识点\第 13 章\系统主界面.avi

MIUI 录音机的主界面十分美观，如图 13-3 所示。



图 13-3　系统主界面

在本节的内容中，将详细讲解 MIUI 录音机主界面的具体实现流程。

13.2.1　实现 UI 布局

系统主界面的 UI 布局文件是 main.xml，具体实现流程如下所示。

（1）设置指定的图片 background.png 为系统主界面的背景图像，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="0dip"
        android:layout_weight="1"
        android:background="@drawable/background"
        android:orientation="vertical" >
```

（2）在屏幕顶部显示计时时间，具体实现代码如下所示。

```
<LinearLayout
    android:id="@+id/time_calculator"
    android:layout_width="fill_parent"
    android:layout_height="0dip"
    android:layout_weight="4"
    android:gravity="center"
    android:orientation="horizontal"
    android:paddingTop="40dip" >
</LinearLayout>
```

（3）使用 FrameLayout 插件实现动态动画效果，首先使用指定图片 tape_bottom.png 生成一个磁带效果，然后使用图片 wheel_left.png 生成左磁带的转动效果，使用图片 wheel_right.png 生成右磁带的转动效果。具体实现代码如下所示。

```
<FrameLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/tape_bottom" />

    <net.micode.soundrecorder.WheelImageView
        android:id="@+id/wheel_left"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left|top"
        android:layout_marginLeft="44dip"
        android:layout_marginTop="60dip"
        android:src="@drawable/wheel_left" />

    <net.micode.soundrecorder.WheelImageView
        android:id="@+id/wheel_right"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|top"
        android:layout_marginRight="45dip"
        android:layout_marginTop="60dip"
        android:src="@drawable/wheel_right" />
```

```
    <net.micode.soundrecorder.WheelImageView
        android:id="@+id/wheel_small_left"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left|bottom"
        android:layout_marginBottom="15dip"
        android:layout_marginLeft="30dip"
        android:src="@drawable/wheel_small_left" />

    <net.micode.soundrecorder.WheelImageView
        android:id="@+id/wheel_small_right"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|bottom"
        android:layout_marginBottom="15dip"
        android:layout_marginRight="30dip"
        android:src="@drawable/wheel_small_right" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/tape_top" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left|top"
        android:layout_marginLeft="55dip"
        android:layout_marginTop="39dip"
        android:orientation="vertical" >

    <LinearLayout
        android:layout_width="0px"
        android:layout_height="0px"
        android:focusable="true"
        android:focusableInTouchMode="true" />

    <net.micode.soundrecorder.RecordNameEditText
        android:id="@+id/file_name"
        android:layout_width="220dip"
        android:layout_height="25dip"
        android:background="#00000000"
        android:selectAllOnFocus="true"
        android:singleLine="true" />
    </LinearLayout>
</FrameLayout>
```

（4）生成屏幕中间的播放进度条效果，在进度条前面显示开始时间“00:00”，在进度条后方显示当前音频的时长。具体实现代码如下所示。

```
<LinearLayout
```

```
    android:layout_width="fill_parent"
    android:layout_height="0dip"
    android:layout_weight="3"
    android:gravity="center"
    android:orientation="vertical" >

    <LinearLayout
        android:id="@+id/vumeter_layout"
        android:layout_width="fill_parent"
        android:layout_height="0dip"
        android:layout_weight="1"
        android:gravity="center"
        android:orientation="horizontal" >
    </LinearLayout>

    <LinearLayout
        android:id="@+id/play_seek_bar_layout"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal" >

    <TextView
        android:id="@+id/starttime"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingLeft="10dip"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:textColor="#7FFFFFFF" />

    <SeekBar
        android:id="@+id/play_seek_bar"
        android:layout_width="0dip"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="horizontal"
        android:paddingLeft="10dip"
        android:paddingRight="10dip" />

    <TextView
        android:id="@+id/totaltime"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingRight="10dip"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:textColor="#7FFFFFFF" />
    </LinearLayout>
  </LinearLayout>
</LinearLayout>
```

执行效果如图 13-4 所示。

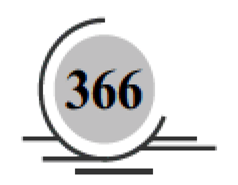




图 13-4　进度条效果

（5）分别生成屏幕底部的播放、暂停、删除、录音等操作按钮，具体实现代码如下所示。

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:background="@drawable/background_key" >

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <ImageButton
            android:id="@+id/newButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@drawable/btn_new" />

        <ImageButton
            android:id="@+id/finishButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@drawable/btn_finish"
            android:visibility="gone" />

        <ImageButton
            android:id="@+id/recordButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@drawable/btn_record" />

        <ImageButton
            android:id="@+id/stopButton"
            android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"
                android:background="@drawable/btn_stop"
                android:visibility="gone" />

            <ImageButton
                android:id="@+id/playButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:background="@drawable/btn_play"
                android:visibility="gone" />

            <ImageButton
                android:id="@+id/pauseButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:background="@drawable/btn_pause"
                android:visibility="gone" />

            <ImageButton
                android:id="@+id/deleteButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:background="@drawable/btn_delete" />
        </LinearLayout>
    </LinearLayout>

</LinearLayout>
```

13.2.2 实现程序文件

系统主界面的程序文件是 SoundRecorder.java，功能是加载 UI 文件中设置的布局控件，监听用户在主界面的操作，并根据操作执行对应的事件处理程序。文件 SoundRecorder.java 的具体实现流程如下所示。

（1）定义系统中需要的常量，并分别设置常量的初始值，具体实现代码如下所示。

```
public class SoundRecorder extends Activity implements Button.OnClickListener,
        Recorder.OnStateChangedListener {
    private static final String TAG = "SoundRecorder";
    private static final String RECORDER_STATE_KEY = "recorder_state";
    private static final String SAMPLE_INTERRUPTED_KEY = "sample_interrupted";
    private static final String MAX_FILE_SIZE_KEY = "max_file_size";
    private static final String AUDIO_3GPP = "audio/3gpp";
    private static final String AUDIO_AMR = "audio/amr";
    private static final String AUDIO_ANY = "audio/*";
    private static final String ANY_ANY = "*/*";
    private static final String FILE_EXTENSION_AMR = ".amr";
    private static final String FILE_EXTENSION_3GPP = ".3gpp";
    public static final int BITRATE_AMR = 2 * 1024 * 8; // bits/sec
    public static final int BITRATE_3GPP = 20 * 1024 * 8; // bits/sec
    private static final int SEEK_BAR_MAX = 10000;
```

```
    private static final long WHEEL_SPEED_NORMAL = 1800;
    private static final long WHEEL_SPEED_FAST = 300;

    private static final long WHEEL_SPEED_SUPER_FAST = 100;

    private static final long SMALL_WHEEL_SPEED_NORMAL = 900;
    private static final long SMALL_WHEEL_SPEED_FAST = 200;
    private static final long SMALL_WHEEL_SPEED_SUPER_FAST = 200;
    private String mRequestedType = AUDIO_ANY;
    private boolean mCanRequestChanged = false;
    private Recorder mRecorder;
    private RecorderReceiver mReceiver;
    private boolean mSampleInterrupted = false;
    private boolean mShowFinishButton = false;
    private String mErrorUiMessage = null;     // Some error messages are displayed
                                               // in the UI, not a dialog. This
                                               // happens when a recording
                                               // is interrupted for some reason.
    private long mMaxFileSize = -1; // can be specified in the intent
    private RemainingTimeCalculator mRemainingTimeCalculator;
    private String mTimerFormat;
    private SoundPool mSoundPool;
    private int mPlaySound;
    private int mPauseSound;
    private HashSet<String> mSavedRecord;
    private long mLastClickTime;
    private int mLastButtonId;
    private final Handler mHandler = new Handler();
    private Runnable mUpdateTimer = new Runnable() {
        public void run() {
            if (!mStopUiUpdate) {
                updateTimerView();
            }
        }
    };
```

（2）在函数 run()中调用函数 updateSeekBar()更新进度条的值，具体实现代码如下所示。

```
private Runnable mUpdateSeekBar = new Runnable() {
    @Override
    public void run() {
        if (!mStopUiUpdate) {
            updateSeekBar();
        }
    }
};
```

（3）根据当前状态来重置底部按钮的状态，例如在播放录音时，底部中间按钮变为停止键，具体实现代码如下所示。

```
@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
```

```
        boolean preShowFinishButton = mShowFinishButton;
        initInternalState(intent);

        if (mShowFinishButton || preShowFinishButton != mShowFinishButton) {
            //重置状态，如果它是一记录请求或状态改变
            mRecorder.reset();
            resetFileNameEditText();
        }
}

private void initInternalState(Intent i) {
        mRequestedType = AUDIO_ANY;
        mShowFinishButton = false;
        if (i != null) {
            String s = i.getType();
            if (AUDIO_AMR.equals(s) || AUDIO_3GPP.equals(s) || AUDIO_ANY.equals(s)
                    || ANY_ANY.equals(s)) {
                mRequestedType = s;
                mShowFinishButton = true;
            } else if (s != null) {
                // we only support amr and 3gpp formats right now
                setResult(RESULT_CANCELED);
                finish();
                return;
            }

            final String EXTRA_MAX_BYTES = android.provider.MediaStore.Audio.Media.EXTRA_MAX_BYTES;
            mMaxFileSize = i.getLongExtra(EXTRA_MAX_BYTES, -1);
        }

        if (AUDIO_ANY.equals(mRequestedType)) {
            mRequestedType = SoundRecorderPreferenceActivity.getRecordType(this);
        } else if (ANY_ANY.equals(mRequestedType)) {
            mRequestedType = AUDIO_3GPP;
        }
}
```

（4）根据系统设置显示当前的状态信息，并且可以保存当前的状态信息，具体实现代码如下所示。

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);

        setContentView(R.layout.main);
        initResourceRefs();
        updateUi(false);
}

@Override
protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
```

```
    if (mRecorder.sampleLength() == 0)
        return;

    Bundle recorderState = new Bundle();

    if (mRecorder.state() != Recorder.RECORDING_STATE) {
        mRecorder.saveState(recorderState);
    }
    recorderState.putBoolean(SAMPLE_INTERRUPTED_KEY, mSampleInterrupted);
    recorderState.putLong(MAX_FILE_SIZE_KEY, mMaxFileSize);

    outState.putBundle(RECORDER_STATE_KEY, recorderState);
}
```

（5）每当重新打开用户界面时，必须重新初始化 UI 控件和视图资源，具体实现代码如下所示。

```
private void initResourceRefs() {
    mNewButton = (ImageButton) findViewById(R.id.newButton);
    mFinishButton = (ImageButton) findViewById(R.id.finishButton);
    mRecordButton = (ImageButton) findViewById(R.id.recordButton);
    mStopButton = (ImageButton) findViewById(R.id.stopButton);
    mPlayButton = (ImageButton) findViewById(R.id.playButton);
    mPauseButton = (ImageButton) findViewById(R.id.pauseButton);
    mDeleteButton = (ImageButton) findViewById(R.id.deleteButton);
    mNewButton.setOnClickListener(this);
    mFinishButton.setOnClickListener(this);
    mRecordButton.setOnClickListener(this);
    mStopButton.setOnClickListener(this);
    mPlayButton.setOnClickListener(this);
    mPauseButton.setOnClickListener(this);
    mDeleteButton.setOnClickListener(this);

    mWheelLeft = (WheelImageView) findViewById(R.id.wheel_left);
    mWheelRight = (WheelImageView) findViewById(R.id.wheel_right);
    mSmallWheelLeft = (WheelImageView) findViewById(R.id.wheel_small_left);
    mSmallWheelRight = (WheelImageView) findViewById(R.id.wheel_small_right);
    mFileNameEditText = (RecordNameEditText) findViewById(R.id.file_name);

    resetFileNameEditText();
    mFileNameEditText.setNameChangeListener(new RecordNameEditText.OnNameChangeListener() {
        @Override
        public void onNameChanged(String name) {
            if (!TextUtils.isEmpty(name)) {
                mRecorder.renameSampleFile(name);
            }
        }
    });

    mTimerLayout = (LinearLayout) findViewById(R.id.time_calculator);
    mVUMeterLayout = (LinearLayout) findViewById(R.id.vumeter_layout);
    mSeekBarLayout = (LinearLayout) findViewById(R.id.play_seek_bar_layout);
    mStartTime = (TextView) findViewById(R.id.starttime);
```

```
        mTotalTime = (TextView) findViewById(R.id.totaltime);
        mPlaySeekBar = (SeekBar) findViewById(R.id.play_seek_bar);
        mPlaySeekBar.setMax(SEEK_BAR_MAX);
        mPlaySeekBar.setOnSeekBarChangeListener(mSeekBarChangeListener);

        mTimerFormat = getResources().getString(R.string.timer_format);

        if (mShowFinishButton) {
            mNewButton.setVisibility(View.GONE);
            mFinishButton.setVisibility(View.VISIBLE);
            mNewButton = mFinishButton; // use mNewButon variable for left
            // button in the control panel
        }

        mSoundPool = new SoundPool(5, AudioManager.STREAM_SYSTEM, 5);
        mPlaySound = mSoundPool.load("/system/media/audio/ui/SoundRecorderPlay.ogg", 1);
        mPauseSound = mSoundPool.load("/system/media/audio/ui/SoundRecorderPause.ogg", 1);

        mLastClickTime = 0;
        mLastButtonId = 0;
}
```

（6）重新设置 EditText 控件中显示的文件名，具体实现代码如下所示。

```
private void resetFileNameEditText() {
        String extension = "";
        if (AUDIO_AMR.equals(mRequestedType)) {
            extension = FILE_EXTENSION_AMR;
        } else if (AUDIO_3GPP.equals(mRequestedType)) {
            extension = FILE_EXTENSION_3GPP;
        }
        mFileNameEditText.initFileName(mRecorder.getRecordDir(), extension, mShowFinishButton);
}
```

（7）分别实现开始录音、停止录音、播放录音、向前动画、向后动画和停止动画效果，具体实现代码如下所示。

```
private void startRecordPlayingAnimation() {
        mWheelLeft.startAnimation(WHEEL_SPEED_NORMAL, true);
        mWheelRight.startAnimation(WHEEL_SPEED_NORMAL, true);
        mSmallWheelLeft.startAnimation(SMALL_WHEEL_SPEED_NORMAL, true);
        mSmallWheelRight.startAnimation(SMALL_WHEEL_SPEED_NORMAL, true);
}

private void stopRecordPlayingAnimation() {
        stopAnimation();
        startRecordPlayingDoneAnimation();
}

private void startRecordPlayingDoneAnimation() {
        mWheelLeft.startAnimation(WHEEL_SPEED_SUPER_FAST, false, 4);
        mWheelRight.startAnimation(WHEEL_SPEED_SUPER_FAST, false, 4);
        mSmallWheelLeft.startAnimation(SMALL_WHEEL_SPEED_SUPER_FAST, false, 2);
        mSmallWheelRight.startAnimation(SMALL_WHEEL_SPEED_SUPER_FAST, false, 2);
```

```
}

private void startForwardAnimation() {
    mWheelLeft.startAnimation(WHEEL_SPEED_FAST, true);
    mWheelRight.startAnimation(WHEEL_SPEED_FAST, true);
    mSmallWheelLeft.startAnimation(SMALL_WHEEL_SPEED_FAST, true);
    mSmallWheelRight.startAnimation(SMALL_WHEEL_SPEED_FAST, true);
}

private void startBackwardAnimation() {
    mWheelLeft.startAnimation(WHEEL_SPEED_FAST, false);
    mWheelRight.startAnimation(WHEEL_SPEED_FAST, false);
    mSmallWheelLeft.startAnimation(SMALL_WHEEL_SPEED_FAST, false);
    mSmallWheelRight.startAnimation(SMALL_WHEEL_SPEED_FAST, false);
}

private void stopAnimation() {
    mWheelLeft.stopAnimation();
    mWheelRight.stopAnimation();
    mSmallWheelLeft.stopAnimation();
    mSmallWheelRight.stopAnimation();
}
```

（8）为了确保不会录上在后台播放的音乐，调用 MediaPlaybackService 暂停后台播放的音频文件，具体实现代码如下所示。

```
private void stopAudioPlayback() {
    // Shamelessly copied from MediaPlaybackService.java, which
    // should be public, but isn't.
    Intent i = new Intent("com.android.music.musicservicecommand");
    i.putExtra("command", "pause");

    sendBroadcast(i);
}
```

（9）监听用户单击底部按钮操作，根据用户单击的按钮执行对应的事件处理程序，例如：

```
public void onClick(View button) {
    if (System.currentTimeMillis() - mLastClickTime < 300) {
        // in order to avoid user click bottom too quickly
        return;
    }

    if (!button.isEnabled())
        return;

    if (button.getId() == mLastButtonId && button.getId() != R.id.newButton) {
            //需要避免开展重复的动作
        return;
    }

    if (button.getId() == R.id.stopButton && System.currentTimeMillis() - mLastClickTime < 1500) {
        //系统崩溃时停止录音
        return;
```

```
        }

        mLastClickTime = System.currentTimeMillis();
        mLastButtonId = button.getId();

        switch (button.getId()) {
            case R.id.newButton:
                mFileNameEditText.clearFocus();
                saveSample();
                mRecorder.reset();
                resetFileNameEditText();
                break;
            case R.id.recordButton:
                showOverwriteConfirmDialogIfConflicts();
                break;
            case R.id.stopButton:
                mRecorder.stop();
                break;
            case R.id.playButton:
                mRecorder.startPlayback(mRecorder.playProgress());
                break;
            case R.id.pauseButton:
                mRecorder.pausePlayback();
                break;
            case R.id.finishButton:
                mRecorder.stop();
                saveSample();
                finish();
                break;
            case R.id.deleteButton:
                showDeleteConfirmDialog();
                break;
        }
}
```

（10）单击录音按钮时通过函数 startRecording()实现录音功能，具体实现代码如下所示。

```
private void startRecording() {
    mRemainingTimeCalculator.reset();
    if (!Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
        mSampleInterrupted = true;
        mErrorUiMessage = getResources().getString(R.string.insert_sd_card);
        updateUi(false);
    } else if (!mRemainingTimeCalculator.diskSpaceAvailable()) {
        mSampleInterrupted = true;
        mErrorUiMessage = getResources().getString(R.string.storage_is_full);
        updateUi(false);
    } else {
        stopAudioPlayback();

        boolean isHighQuality = SoundRecorderPreferenceActivity.isHighQuality(this);
        if (AUDIO_AMR.equals(mRequestedType)) {
```

```
            mRemainingTimeCalculator.setBitRate(BITRATE_AMR);
            int outputfileformat = isHighQuality ? MediaRecorder.OutputFormat.AMR_WB
                    : MediaRecorder.OutputFormat.AMR_NB;
            mRecorder.startRecording(outputfileformat, mFileNameEditText.getText().toString(),
                    FILE_EXTENSION_AMR, isHighQuality, mMaxFileSize);
        } else if (AUDIO_3GPP.equals(mRequestedType)) {
            // HACKME: for HD2, there is an issue with high quality 3gpp
            // use low quality instead
            if (Build.MODEL.equals("HTC HD2")) {
                isHighQuality = false;
            }

            mRemainingTimeCalculator.setBitRate(BITRATE_3GPP);
            mRecorder.startRecording(MediaRecorder.OutputFormat.THREE_GPP, mFileNameEditText
                    .getText().toString(), FILE_EXTENSION_3GPP, isHighQuality, mMaxFileSize);
        } else {
            throw new IllegalArgumentException("Invalid output file type requested");
        }

        if (mMaxFileSize != -1) {
            mRemainingTimeCalculator.setFileSizeLimit(mRecorder.sampleFile(), mMaxFileSize);
        }
    }
}
```

（11）通过函数 onResume()实现重置按钮的处理功能，具体实现代码如下所示。

```
@Override
protected void onResume() {
    super.onResume();
    String type = SoundRecorderPreferenceActivity.getRecordType(this);
    if (mCanRequestChanged && !TextUtils.equals(type, mRequestedType)) {
        saveSample();
        mRecorder.reset();
        mRequestedType = type;
        resetFileNameEditText();
    }
    mCanRequestChanged = false;

    if (!mRecorder.syncStateWithService()) {
        mRecorder.reset();
        resetFileNameEditText();
    }

    if (mRecorder.state() == Recorder.RECORDING_STATE) {
        String preExtension = AUDIO_AMR.equals(mRequestedType) ? FILE_EXTENSION_AMR
                : FILE_EXTENSION_3GPP;
        if (!mRecorder.sampleFile().getName().endsWith(preExtension)) {
            // the extension is changed need to stop current recording
            mRecorder.reset();
            resetFileNameEditText();
        } else {
```

```
                // restore state
                if (!mShowFinishButton) {
                    String fileName = mRecorder.sampleFile().getName().replace(preExtension, "");
                    mFileNameEditText.setText(fileName);
                }

                if (AUDIO_AMR.equals(mRequestedType)) {
                    mRemainingTimeCalculator.setBitRate(BITRATE_AMR);
                } else if (AUDIO_3GPP.equals(mRequestedType)) {
                    mRemainingTimeCalculator.setBitRate(BITRATE_3GPP);
                }
            }
        } else {
            File file = mRecorder.sampleFile();
            if (file != null && !file.exists()) {
                mRecorder.reset();
                resetFileNameEditText();
            }
        }

        IntentFilter filter = new IntentFilter();
        filter.addAction(RecorderService.RECORDER_SERVICE_BROADCAST_NAME);
        registerReceiver(mReceiver, filter);

        mStopUiUpdate = false;
        updateUi(true);

        if (RecorderService.isRecording()) {
            Intent intent = new Intent(this, RecorderService.class);
            intent.putExtra(RecorderService.ACTION_NAME,
                    RecorderService.ACTION_DISABLE_MONITOR_REMAIN_TIME);
            startService(intent);
        }
}
```

（12）通过函数 onPause()实现暂停按钮的事件处理功能，具体实现代码如下所示。

```
@Override
protected void onPause() {
    if (mRecorder.state() != Recorder.RECORDING_STATE || mShowFinishButton
            || mMaxFileSize != -1) {
        mRecorder.stop();
        saveSample();
        mFileNameEditText.clearFocus();
        ((NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE))
                .cancel(RecorderService.NOTIFICATION_ID);
    }

    if (mReceiver != null) {
        unregisterReceiver(mReceiver);
    }
```

```
        mCanRequestChanged = true;
        mStopUiUpdate = true;
        stopAnimation();

        if (RecorderService.isRecording()) {
            Intent intent = new Intent(this, RecorderService.class);
            intent.putExtra(RecorderService.ACTION_NAME,
                    RecorderService.ACTION_ENABLE_MONITOR_REMAIN_TIME);
            startService(intent);
        }

        super.onPause();
    }

    @Override
    protected void onStop() {
        if (mShowFinishButton) {
            finish();
        }
        super.onStop();
    }
```

（13）如果录音完毕，通过函数 saveSample()实现单击停止录音按钮的事件处理程序，保存当前的音频文件到媒体库，具体实现代码如下所示。

```
    private void saveSample() {
        if (mRecorder.sampleLength() == 0)
            return;
        if (!mSavedRecord.contains(mRecorder.sampleFile().getAbsolutePath())) {
            Uri uri = null;
            try {
                uri = this.addToMediaDB(mRecorder.sampleFile());
            } catch (UnsupportedOperationException ex) { // Database
                // manipulation
                // failure
                return;
            }
            if (uri == null) {
                return;
            }
            mSavedRecord.add(mRecorder.sampleFile().getAbsolutePath());
            setResult(RESULT_OK, new Intent().setData(uri));
        }
    }
```

（14）通过函数 showDeleteConfirmDialog()实现单击删除按钮的事件处理程序，弹出一个确认删除对话框，具体实现代码如下所示。

```
    private void showDeleteConfirmDialog() {
        AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);
        dialogBuilder.setIcon(android.R.drawable.ic_dialog_alert);
        dialogBuilder.setTitle(R.string.delete_dialog_title);
        dialogBuilder.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
            @Override
```

```
            public void onClick(DialogInterface dialog, int which) {
                mRecorder.delete();
            }
        });
    dialogBuilder.setNegativeButton(android.R.string.cancel,
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    mLastButtonId = 0;
                }
            });
    dialogBuilder.show();
}
```

（15）函数 showOverwriteConfirmDialogIfConflicts()的功能是，在保存录音文件时如果在媒体库中已经存在同名文件则会弹出一个提示对话框，具体实现代码如下所示。

```
private void showOverwriteConfirmDialogIfConflicts() {
    String fileName = mFileNameEditText.getText().toString()
            + (AUDIO_AMR.equals(mRequestedType) ? FILE_EXTENSION_AMR : FILE_EXTENSION_
            3GPP);

    if (mRecorder.isRecordExisted(fileName) && !mShowFinishButton) {
        AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);
        dialogBuilder.setIcon(android.R.drawable.ic_dialog_alert);
        dialogBuilder.setTitle(getString(R.string.overwrite_dialog_title, fileName));
        dialogBuilder.setPositiveButton(android.R.string.ok,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        startRecording();
                    }
                });
        dialogBuilder.setNegativeButton(android.R.string.cancel,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        mLastButtonId = 0;
                    }
                });
        dialogBuilder.show();
    } else {
        startRecording();
    }
}
```

（16）通过函数 addToPlaylist()将录制音频添加到播放列表中，具体实现代码如下所示。

```
private void addToPlaylist(ContentResolver resolver, int audioId, long playlistId) {
    String[ ] cols = new String[ ] {
        "count(*)"
    };
    Uri uri = MediaStore.Audio.Playlists.Members.getContentUri("external", playlistId);
    Cursor cur = resolver.query(uri, cols, null, null, null);
```

```
        cur.moveToFirst();
        final int base = cur.getInt(0);
        cur.close();
        ContentValues values = new ContentValues();
        values.put(MediaStore.Audio.Playlists.Members.PLAY_ORDER, Integer.valueOf(base + audioId));
        values.put(MediaStore.Audio.Playlists.Members.AUDIO_ID, audioId);
        resolver.insert(uri, values);
}
```

（17）通过函数 getPlaylistId()从表 audio_playlists 中获取默认播放列表的 ID，具体实现代码如下所示。

```
private int getPlaylistId(Resources res) {
        Uri uri = MediaStore.Audio.Playlists.getContentUri("external");
        final String[ ] ids = new String[ ] {
                MediaStore.Audio.Playlists._ID
        };
        final String where = MediaStore.Audio.Playlists.NAME + "=?";
        final String[ ] args = new String[ ] {
                res.getString(R.string.audio_db_playlist_name)
        };
        Cursor cursor = query(uri, ids, where, args, null);
        if (cursor == null) {
                Log.v(TAG, "query returns null");
        }
        int id = -1;
        if (cursor != null) {
                cursor.moveToFirst();
                if (!cursor.isAfterLast()) {
                        id = cursor.getInt(0);
                }
                cursor.close();
        }
        return id;
}
```

（18）通过函数 createPlaylist()创建一个播放列表，前提是系统中没有同名列表存在，具体实现代码如下所示。

```
private Uri createPlaylist(Resources res, ContentResolver resolver) {
        ContentValues cv = new ContentValues();
        cv.put(MediaStore.Audio.Playlists.NAME, res.getString(R.string.audio_db_playlist_name));
        Uri uri = resolver.insert(MediaStore.Audio.Playlists.getContentUri("external"), cv);
        if (uri == null) {
                new AlertDialog.Builder(this).setTitle(R.string.app_name)
                                .setMessage(R.string.error_mediadb_new_record)
                                .setPositiveButton(R.string.button_ok, null).setCancelable(false).show();
        }
        return uri;
}
```

（19）函数 getTimerImage()的功能是根据当前时间调用对应的图片，以实现图形化的数字时钟效果，具体实现代码如下所示。

```
private ImageView getTimerImage(char number) {
```

```
    ImageView image = new ImageView(this);
    LayoutParams lp = new LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
    if (number != ':') {
        image.setBackgroundResource(R.drawable.background_number);
    }
    switch (number) {
        case '0':
            image.setImageResource(R.drawable.number_0);
            break;
        case '1':
            image.setImageResource(R.drawable.number_1);
            break;
        case '2':
            image.setImageResource(R.drawable.number_2);
            break;
        case '3':
            image.setImageResource(R.drawable.number_3);
            break;
        case '4':
            image.setImageResource(R.drawable.number_4);
            break;
        case '5':
            image.setImageResource(R.drawable.number_5);
            break;
        case '6':
            image.setImageResource(R.drawable.number_6);
            break;
        case '7':
            image.setImageResource(R.drawable.number_7);
            break;
        case '8':
            image.setImageResource(R.drawable.number_8);
            break;
        case '9':
            image.setImageResource(R.drawable.number_9);
            break;
        case ':':
            image.setImageResource(R.drawable.colon);
            break;
    }
    image.setLayoutParams(lp);
    return image;
}
```

（20）更新 MM：SS 格式的计时器的时间，如果正在播放音频，也随之更新进度条的时间，具体实现代码如下所示。

```
private void updateTimerView() {
    int state = mRecorder.state();
    boolean ongoing = state == Recorder.RECORDING_STATE || state == Recorder.PLAYING_STATE;

    long time = mRecorder.progress();
```

```
        String timeStr = String.format(mTimerFormat, time / 60, time % 60);
        mTimerLayout.removeAllViews();
        for (int i = 0; i < timeStr.length(); i++) {
            mTimerLayout.addView(getTimerImage(timeStr.charAt(i)));
        }

        if (state == Recorder.RECORDING_STATE) {
            updateTimeRemaining();
        }

        if (ongoing) {
            mHandler.postDelayed(mUpdateTimer, 500);
        }
    }

    private void setTimerView(float progress) {
        long time = (long) (progress * mRecorder.sampleLength());
        String timeStr = String.format(mTimerFormat, time / 60, time % 60);
        mTimerLayout.removeAllViews();
        for (int i = 0; i < timeStr.length(); i++) {
            mTimerLayout.addView(getTimerImage(timeStr.charAt(i)));
        }
    }

    private void updateSeekBar() {
        if (mRecorder.state() == Recorder.PLAYING_STATE) {
            mPlaySeekBar.setProgress((int) (SEEK_BAR_MAX * mRecorder.playProgress()));
            mHandler.postDelayed(mUpdateSeekBar, 10);
        }
    }
```

（21）函数 updateTimeRemaining()的功能比较人性化，此函数在录音状态下被调用，用于显示还可以录制多少时间。如果还可以录制 5 分钟，则在 UI 中显示一个倒计时时钟，如果已经用完这 5 分钟，则立即停止录制，具体实现代码如下所示。

```
    private void updateTimeRemaining() {
        long t = mRemainingTimeCalculator.timeRemaining();

        if (t <= 0) {
            mSampleInterrupted = true;

            int limit = mRemainingTimeCalculator.currentLowerLimit();
            switch (limit) {
                case RemainingTimeCalculator.DISK_SPACE_LIMIT:
                    mErrorUiMessage = getResources().getString(R.string.storage_is_full);
                    break;
                case RemainingTimeCalculator.FILE_SIZE_LIMIT:
                    mErrorUiMessage = getResources().getString(R.string.max_length_reached);
                    break;
                default:
                    mErrorUiMessage = null;
```

```
                    break;
            }

            mRecorder.stop();
            return;
        }
}
```

（22）根据用户的操作和当前的状态更新 UI 界面视图，具体实现代码如下所示。

```
private void updateUi(boolean skipRewindAnimation) {
    switch (mRecorder.state()) {
        case Recorder.IDLE_STATE:
            mLastButtonId = 0;
        case Recorder.PLAYING_PAUSED_STATE:
            if (mRecorder.sampleLength() == 0) {
                mNewButton.setEnabled(true);
                mNewButton.setVisibility(View.VISIBLE);
                mRecordButton.setVisibility(View.VISIBLE);
                mStopButton.setVisibility(View.GONE);
                mPlayButton.setVisibility(View.GONE);
                mPauseButton.setVisibility(View.GONE);
                mDeleteButton.setEnabled(false);
                mRecordButton.requestFocus();

                mVUMeterLayout.setVisibility(View.VISIBLE);
                mSeekBarLayout.setVisibility(View.GONE);
            } else {
                mNewButton.setEnabled(true);
                mNewButton.setVisibility(View.VISIBLE);
                mRecordButton.setVisibility(View.GONE);
                mStopButton.setVisibility(View.GONE);
                mPlayButton.setVisibility(View.VISIBLE);
                mPauseButton.setVisibility(View.GONE);
                mDeleteButton.setEnabled(true);
                mPauseButton.requestFocus();

                mVUMeterLayout.setVisibility(View.GONE);
                mSeekBarLayout.setVisibility(View.VISIBLE);
                mStartTime.setText(String.format(mTimerFormat, 0, 0));
                mTotalTime.setText(String.format(mTimerFormat, mRecorder.sampleLength() / 60,
                        mRecorder.sampleLength() % 60));
            }
            mFileNameEditText.setEnabled(true);
            mFileNameEditText.clearFocus();

            if (mRecorder.sampleLength() > 0) {
                if (mRecorder.state() == Recorder.PLAYING_PAUSED_STATE) {
                    stopAnimation();
                    if (SoundRecorderPreferenceActivity.isEnabledSoundEffect(this)) {
                        mSoundPool.play(mPauseSound, 1.0f, 1.0f, 0, 0, 1);
                    }
                } else {
```

```
                mPlaySeekBar.setProgress(0);
                if (!skipRewindAnimation) {
                    stopRecordPlayingAnimation();
                } else {
                    stopAnimation();
                }
            }
        } else {
            stopAnimation();
        }

        // we allow only one toast at one time
        if (mSampleInterrupted && mErrorUiMessage == null) {
            Toast.makeText(this, R.string.recording_stopped, Toast.LENGTH_SHORT).show();
        }

        if (mErrorUiMessage != null) {
            Toast.makeText(this, mErrorUiMessage, Toast.LENGTH_SHORT).show();
        }

        break;
    case Recorder.RECORDING_STATE:
        mNewButton.setEnabled(false);
        mNewButton.setVisibility(View.VISIBLE);
        mRecordButton.setVisibility(View.GONE);
        mStopButton.setVisibility(View.VISIBLE);
        mPlayButton.setVisibility(View.GONE);
        mPauseButton.setVisibility(View.GONE);
        mDeleteButton.setEnabled(false);
        mStopButton.requestFocus();

        mVUMeterLayout.setVisibility(View.VISIBLE);
        mSeekBarLayout.setVisibility(View.GONE);

        mFileNameEditText.setEnabled(false);

        startRecordPlayingAnimation();
        mPreviousVUMax = 0;
        break;

    case Recorder.PLAYING_STATE:
        mNewButton.setEnabled(false);
        mNewButton.setVisibility(View.VISIBLE);
        mRecordButton.setVisibility(View.GONE);
        mStopButton.setVisibility(View.GONE);
        mPlayButton.setVisibility(View.GONE);
        mPauseButton.setVisibility(View.VISIBLE);
        mDeleteButton.setEnabled(false);
        mPauseButton.requestFocus();

        mVUMeterLayout.setVisibility(View.GONE);
        mSeekBarLayout.setVisibility(View.VISIBLE);
```

```
            mFileNameEditText.setEnabled(false);

            if (SoundRecorderPreferenceActivity.isEnabledSoundEffect(this)) {
                mSoundPool.play(mPlaySound, 1.0f, 1.0f, 0, 0, 1);
            }
            startRecordPlayingAnimation();
            break;
    }

    updateTimerView();
    updateSeekBar();
    updateVUMeterView();
}
```

（23）当 MediaPlayer 出错时会调用 onError()函数，具体实现代码如下所示。

```
public void onError(int error) {
    Resources res = getResources();
    String message = null;
    switch (error) {
        case Recorder.STORAGE_ACCESS_ERROR:
            message = res.getString(R.string.error_sdcard_access);
            break;
        case Recorder.IN_CALL_RECORD_ERROR:
            // TODO: update error message to reflect that the recording
            // could not be
            // performed during a call.
        case Recorder.INTERNAL_ERROR:
            message = res.getString(R.string.error_app_internal);
            break;
    }
    if (message != null) {
        new AlertDialog.Builder(this).setTitle(R.string.app_name).setMessage(message)
                .setPositiveButton(R.string.button_ok, null).setCancelable(false).show();
    }
}
```

13.3　系统设置界面

知识点讲解：光盘:视频\知识点\第 13 章\系统设置界面.avi

当单击设备的 MENU 键时会弹出一个操作选项框，如图 13-5 所示。

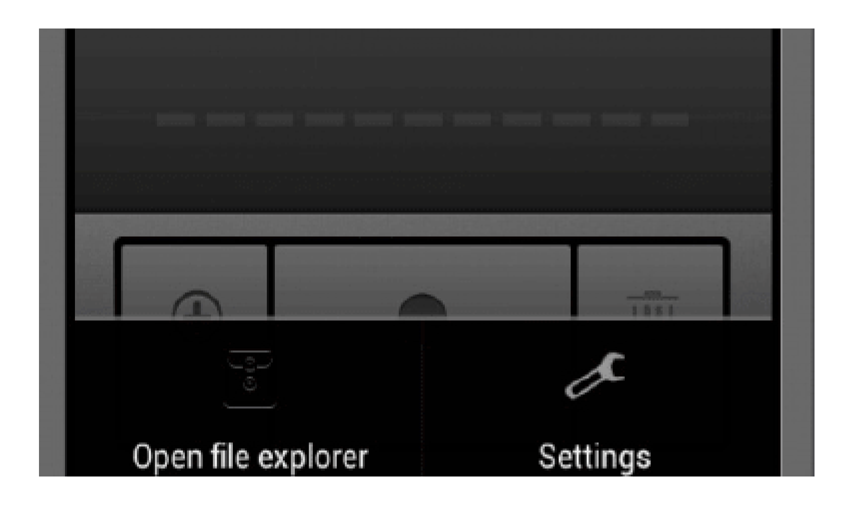


图 13-5　操作选项框

当选择选项框中的 Open file explorer 选项时会打开文件浏览器。在本节的内容中，将详细讲解系统设置界面的具体实现流程。

13.3.1　事件处理程序

当用户单击图 13-5 中的某个选项时，会调用函数 onOptionsItemSelected()来监听操作，并执行对应的事件处理程序，具体实现代码如下所示。

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
        menu.clear();
        if (mRecorder.state() == Recorder.RECORDING_STATE
                || mRecorder.state() == Recorder.PLAYING_STATE) {
            return false;
        } else {
            getMenuInflater().inflate(R.layout.view_list_menu, menu);
            return true;
        }
}
public boolean onOptionsItemSelected(MenuItem item) {
        Intent intent;
        switch (item.getItemId()) {
            case R.id.menu_fm:
                saveSample();
                intent = new Intent();
                intent.addCategory(Intent.CATEGORY_DEFAULT);
                intent.setData(Uri.parse("file://" + mRecorder.getRecordDir()));
                startActivity(intent);
                break;
            case R.id.menu_setting:
                intent = new Intent(this, SoundRecorderPreferenceActivity.class);
                startActivity(intent);
                break;
            default:
                break;
        }
        return true;
}
```

13.3.2　实现程序文件

系统设置界面的布局文件是 preferences.xml，具体实现代码如下所示。

```
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory>
        <ListPreference
            android:key="pref_key_record_type"
            android:title="@string/pref_title_record_type"
            android:entries="@array/prefEntries_recordType"
            android:entryValues="@array/prefValues_recordType"
```

```
            android:dialogTitle="@string/prefDialogTitle_recordType"
            android:defaultValue="@string/prefDefault_recordType" />
        <CheckBoxPreference
            android:key="pref_key_enable_high_quality"
            android:title="@string/pref_title_enable_high_quality"
            android:summary="@string/pref_summary_enable_high_quality"
            android:defaultValue="true" />
    </PreferenceCategory>
    <PreferenceCategory>
        <CheckBoxPreference
            android:key="pref_key_enable_sound_effect"
            android:title="@string/pref_title_enable_sound_effect"
            android:summary="@string/pref_summary_enable_sound_effect"
            android:defaultValue="true" />
    </PreferenceCategory>
</PreferenceScreen>
```

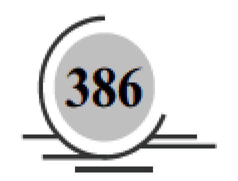
对应的程序文件是 SoundRecorderPreferenceActivity.java，用于加载显示 UI 布局文件中的控件，并监听用户的操作选项对系统进行设置。文件 SoundRecorderPreferenceActivity.java 的具体实现代码如下所示。

```
public class SoundRecorderPreferenceActivity extends PreferenceActivity {
    private static final String RECORD_TYPE = "pref_key_record_type";
    private static final String ENABLE_HIGH_QUALITY = "pref_key_enable_high_quality";
    private static final String ENABLE_SOUND_EFFECT = "pref_key_enable_sound_effect";
    @Override
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        addPreferencesFromResource(R.xml.preferences);
    }
    public static String getRecordType(Context context) {
        SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(context);
        return settings.getString(RECORD_TYPE, context.getString(R.string.prefDefault_recordType));
    }
    public static boolean isHighQuality(Context context) {
        SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(context);
        return settings.getBoolean(ENABLE_HIGH_QUALITY, true);
    }
    public static boolean isEnabledSoundEffect(Context context) {
        SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(context);
        return settings.getBoolean(ENABLE_SOUND_EFFECT, true);
    }
}
```

系统设置界面的执行效果如图 13-6 所示。



图 13-6　系统设置界面



13.4　修改文本框的文本

知识点讲解：光盘:视频\知识点\第 13 章\修改文本框的文本.avi

在系统主界面的 EditText 中，可以设置修改录音文件的名字，此功能是通过文件 RecordNameEditText.java 实现的，具体实现代码如下所示。

```
public class RecordNameEditText extends EditText {
    private Context mContext;
    private InputMethodManager mInputMethodManager;
    private OnNameChangeListener mNameChangeListener;
    private String mDir;
    private String mExtension;
    private String mOriginalName;
    public interface OnNameChangeListener {
        void onNameChanged(String name);
    }
    public RecordNameEditText(Context context) {
        super(context, null);
        mContext = context;
        mInputMethodManager = (InputMethodManager) context
                .getSystemService(Context.INPUT_METHOD_SERVICE);
        mNameChangeListener = null;
    }
    public RecordNameEditText(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        mInputMethodManager = (InputMethodManager) context
                .getSystemService(Context.INPUT_METHOD_SERVICE);
        mNameChangeListener = null;
    }
    public RecordNameEditText(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        mContext = context;
        mInputMethodManager = (InputMethodManager) context
                .getSystemService(Context.INPUT_METHOD_SERVICE);
        mNameChangeListener = null;
    }
    public void setNameChangeListener(OnNameChangeListener listener) {
        mNameChangeListener = listener;
    }
    public void initFileName(String dir, String extension, boolean englishOnly) {
        mDir = dir;
        mExtension = extension;
        //初始化默认名字
        if (!englishOnly) {
            setText(getProperFileName(mContext.getString(R.string.default_record_name)));
        } else {
```

```
            SimpleDateFormat dataFormat = new SimpleDateFormat("MMddHHmmss");
            setText(getProperFileName("rec_" + dataFormat.format(Calendar.getInstance().getTime())));
        }
    }
    private String getProperFileName(String name) {
        String uniqueName = name;

        if (isFileExisted(uniqueName)) {
            int i = 2;
            while (true) {
                String temp = uniqueName + "(" + i + ")";
                if (!isFileExisted(temp)) {
                    uniqueName = temp;
                    break;
                }
                i++;
            }
        }
        return uniqueName;
    }
    private boolean isFileExisted(String name) {
        String fullName = mDir + "/" + name.trim() + mExtension;
        File file = new File(fullName);
        return file.exists();
    }
    @Override
    public boolean onKeyUp(int keyCode, KeyEvent event) {
        switch (keyCode) {
            case KeyEvent.KEYCODE_ENTER:
                if (mNameChangeListener != null) {
                    String name = getText().toString().trim();
                    if (!TextUtils.isEmpty(name)) {
                        //使用新名字
                        setText(name);
                        mNameChangeListener.onNameChanged(name);
                    } else {
                        //使用原始名字
                        setText(mOriginalName);
                    }
                    clearFocus();
                    //隐藏键盘
                    mInputMethodManager.hideSoftInputFromWindow(getWindowToken(), 0);
                    return true;
                }
                break;
            default:
                break;
        }
        return super.onKeyUp(keyCode, event);
    }
    @Override
```

```
    protected void onFocusChanged(boolean focused, int direction, Rect previouslyFocusedRect) {
        super.onFocusChanged(focused, direction, previouslyFocusedRect);
        if (!focused && mNameChangeListener != null) {
            String name = getText().toString().trim();
            if (!TextUtils.isEmpty(name)) {
                //使用新名字
                setText(name);
                mNameChangeListener.onNameChanged(name);
            } else {
                //使用原始名字
                setText(mOriginalName);
            }
            // hide the keyboard
            mInputMethodManager.hideSoftInputFromWindow(getWindowToken(), 0);
        } else if (focused) {
            mOriginalName = getText().toString();
        }
    }
}
```

13.5　计算剩余时间

知识点讲解：光盘:视频\知识点\第 13 章\计算剩余时间.avi

在系统主界面中会显示当前存储卡的容量还允许录制多少时间。如果还可以录制 5 分钟，则在 UI 中显示一个倒计时时钟，如果已经用完这 5 分钟，则立即停止录制。计算剩余时间的核心功能是通过文件 RemainingTimeCalculator.java 实现的，具体实现代码如下所示。

```
public class RemainingTimeCalculator {
    public static final int UNKNOWN_LIMIT = 0;
    public static final int FILE_SIZE_LIMIT = 1;
    public static final int DISK_SPACE_LIMIT = 2;
    private static final int EXTERNAL_STORAGE_BLOCK_THREADHOLD = 32;
    private int mCurrentLowerLimit = UNKNOWN_LIMIT;
    private File mRecordingFile;
    private long mMaxBytes;
    private int mBytesPerSecond;
    private long mBlocksChangedTime;
    private long mLastBlocks;
    private long mFileSizeChangedTime;
    private long mLastFileSize;
    public RemainingTimeCalculator() {
    }
    /**
     * 设置文件大小限制
     */
    public void setFileSizeLimit(File file, long maxBytes) {
        mRecordingFile = file;
        mMaxBytes = maxBytes;
```

```
}
/**
 * 重置 interpolation
 */
public void reset() {
    mCurrentLowerLimit = UNKNOWN_LIMIT;
    mBlocksChangedTime = -1;
    mFileSizeChangedTime = -1;
}
/**
 *返回的时间（以秒计），我们可以继续录制
 */
public long timeRemaining() {
    // Calculate how long we can record based on free disk space
    StatFs fs = null;
    long blocks = -1;
    long blockSize = -1;
    long now = System.currentTimeMillis();

    fs = new StatFs(Environment.getExternalStorageDirectory().getAbsolutePath());
    blocks = fs.getAvailableBlocks() - EXTERNAL_STORAGE_BLOCK_THREADHOLD;
    blockSize = fs.getBlockSize();
    if (blocks < 0) {
        blocks = 0;
    }

    if (mBlocksChangedTime == -1 || blocks != mLastBlocks) {
        mBlocksChangedTime = now;
        mLastBlocks = blocks;
    }
    // at mBlocksChangedTime we had this much time
    long result = mLastBlocks * blockSize / mBytesPerSecond;
    //现在我们有这么多的时间
    result -= (now - mBlocksChangedTime) / 1000;
    if (mRecordingFile == null) {
        mCurrentLowerLimit = DISK_SPACE_LIMIT;
        return result;
    }
    //计算预估
    mRecordingFile = new File(mRecordingFile.getAbsolutePath());
    long fileSize = mRecordingFile.length();
    if (mFileSizeChangedTime == -1 || fileSize != mLastFileSize) {
        mFileSizeChangedTime = now;
        mLastFileSize = fileSize;
    }
    long result2 = (mMaxBytes - fileSize) / mBytesPerSecond;
    result2 -= (now - mFileSizeChangedTime) / 1000;
    result2 -= 1; // just for safety
    mCurrentLowerLimit = result < result2 ? DISK_SPACE_LIMIT : FILE_SIZE_LIMIT;
    return Math.min(result, result2);
}
```

```
    /**
     * 指出限制
     */
    public int currentLowerLimit() {
        return mCurrentLowerLimit;
    }
    /**
     *是否有任何一点的尝试开始录制
     */
    public boolean diskSpaceAvailable() {
        StatFs fs = new StatFs(Environment.getExternalStorageDirectory().getAbsolutePath());
        // keep one free block
        return fs.getAvailableBlocks() > EXTERNAL_STORAGE_BLOCK_THREADHOLD;
    }
    /**
     * 设置 interpolation 的比特率
     *
     * @param bitRate the bit rate to set in bits/sec.
     */
    public void setBitRate(int bitRate) {
        mBytesPerSecond = bitRate / 8;
    }
}
```

13.6 素材修饰

知识点讲解：光盘:视频\知识点\第 13 章\素材修饰.avi

为了使系统中的 UI 控件以完美的效果展示出来，MIUI 团队对图片和动画实现了修饰美化。其中使用文件 SeamlessAnimation.java 实现了无缝动画显示效果，具体实现代码如下所示。

```
public class SeamlessAnimation extends Animation {
    private float mFromDegrees;
    private float mToDegrees;
    private float mPivotX;
    private float mPivotY;
    private int mPivotXType;
    private float mPivotXValue;
    private int mPivotYType;
    private float mPivotYValue;
    private boolean mCancelled;
    private float mDegree;
    public SeamlessAnimation(float fromDegrees, float toDegrees, int pivotXType, float pivotXValue,
            int pivotYType, float pivotYValue) {
        mFromDegrees = fromDegrees;
        mToDegrees = toDegrees;
        mPivotXType = pivotXType;
        mPivotXValue = pivotXValue;
        mPivotYType = pivotYType;
```

```
        mPivotYValue = pivotYValue;
        mCancelled = false;
        mDegree = fromDegrees;
    }
    public void initialize(int width, int height, int parentWidth, int parentHeight) {
        super.initialize(width, height, parentWidth, parentHeight);
        mPivotX = resolveSize(mPivotXType, mPivotXValue, width, parentWidth);
        mPivotY = resolveSize(mPivotYType, mPivotYValue, height, parentHeight);
    }
    public float getDegree() {
        return mDegree;
    }
    @Override
    public void cancel() {
        super.cancel();
        mCancelled = true;
    }
    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t) {
        if (!mCancelled) {
            mDegree = mFromDegrees + ((mToDegrees - mFromDegrees) * interpolatedTime);
        }
        t.getMatrix().setRotate(mDegree, mPivotX, mPivotY);
    }
}
```

通过文件 WheelImageView.java 实现了磁带轮子图片和无缝动画效果的完美融合，具体实现代码如下所示。

```
public class WheelImageView extends ImageView {
    SeamlessAnimation mAnimation;
    public WheelImageView(Context context) {
        super(context);
        mAnimation = null;
    }
    public WheelImageView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mAnimation = null;
    }
    public WheelImageView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        mAnimation = null;
    }
    private void initAnimation(long duration, boolean isForward, int repeatCount) {
        LinearInterpolator lir = new LinearInterpolator();
        float from;
        float to;
        if (isForward) {
            from = (mAnimation == null || mAnimation.getRepeatCount() != Animation.INFINITE) ? 0.0f
                    : mAnimation.getDegree();
            to = from + 360.0f;
        } else {
```

```
                from = (mAnimation == null || mAnimation.getRepeatCount() != Animation.INFINITE) ? 360.0f
                        : mAnimation.getDegree();
                to = from - 360.0f;
            }
            if (isForward) {
                mAnimation = new SeamlessAnimation(from, to, Animation.RELATIVE_TO_SELF, 0.5f,
                        Animation.RELATIVE_TO_SELF, 0.5f);
            } else {
                mAnimation = new SeamlessAnimation(from, to, Animation.RELATIVE_TO_SELF, 0.5f,
                        Animation.RELATIVE_TO_SELF, 0.5f);
            }
            mAnimation.setDuration(duration);
            mAnimation.setRepeatMode(Animation.RESTART);
            mAnimation.setRepeatCount(repeatCount);
            mAnimation.setInterpolator(lir);
        }
        public void startAnimation(long duration, boolean isForward) {
            startAnimation(duration, isForward, Animation.INFINITE);
        }
        public void startAnimation(long duration, boolean isForward, int repeatCount) {
            initAnimation(duration, isForward, repeatCount);
            startAnimation(mAnimation);
        }
        public void stopAnimation() {
            if (mAnimation != null) {
                mAnimation.cancel();
                mAnimation = null;
                clearAnimation();
            }
        }
}
```

第 14 章　智能楼宇灯光控制系统

在本章的实例项目中，将演示在 Android 设备中开发一个智能楼宇灯光控制系统的方法。本实例是一个商业项目，需要开发人员额外编写驱动程序和底层蓝牙控制程序。在本章的内容中，将只讲解 Android 应用程序的实现过程。本实例为读者进行智能家居系统开发提供了很好的参考资料和素材，希望大家认真学习。

14.1　布 局 文 件

知识点讲解：光盘:视频\知识点\第 14 章\布局文件.avi

布局文件即 UI 界面设计文件，用于规划在屏幕中显示的控件、图像和文本元素。在本节的内容中，将首先详细讲解组界面布局文件的实现过程。

14.1.1　主布局文件

编写主布局文件 main.xml，设置屏幕中间显示动态显示内容，屏幕底部为固定的界面，具体实现代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:custom="http://www.javaeye.com/custom"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/title_relativeLayout"
    android:background="@drawable/one">
     <!-- 中间动态显示界面 -->
     <ViewFlipper android:id="@+id/fliper"
          android:layout_width="fill_parent"
          android:layout_height="fill_parent"
          android:layout_below="@id/title_relativeLayout"
          android:background="#0000"
          android:layout_marginBottom="50.0dip">

     </ViewFlipper>

     <!-- 底部为固定的布局 -->
          <!-- 底部为固定的布局 -->
     <RelativeLayout android:orientation="horizontal"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_alignParentBottom="true"
```

```
            style="@android:style/ButtonBar"
            android:background="@drawable/title_background">
        <ImageButton android:background="#0000" android:layout_width="wrap_content"
 android:src="@drawable/add" android:id="@+id/imageButton1" android:layout_height="wrap_content"
 android:layout_alignParentBottom="true" android:layout_alignParentLeft="true"></ImageButton>
        <ImageButton android:layout_height="wrap_content" android:background="#0000"
 android:id="@+id/imageButton2" android:layout_width="wrap_content" android:src="@drawable/menu"
 android:layout_alignTop="@+id/imageButton1" android:layout_alignParentRight="true"></ImageButton>


        </RelativeLayout>
        </RelativeLayout>
```

14.1.2　实现蓝牙控制界面

编写文件 bluetooth.xml，通过按钮控件、ImageView 控件和 ToggleButton 控件实现蓝牙控制界面，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/one">
      <RelativeLayout android:layout_width="match_parent" android:layout_height="match_parent"
android:id="@+id/relativeLayout1">
            <ListView android:id="@+id/lvDevices" android:layout_height="wrap_content"
android:layout_width="match_parent" android:layout_alignParentLeft="true"
android:layout_alignParentBottom="true" android:layout_below="@+id/btnExit"></ListView>
            <Button android:text="Native Bluetooth visibility" android:layout_height="wrap_content"
android:layout_width="wrap_content" android:textSize="25sp" android:id="@+id/btnDis"
android:layout_alignParentTop="true" android:layout_alignRight="@+id/imageView1"
android:layout_marginRight="64dp" android:layout_marginTop="315dp"></Button>
            <Button android:layout_width="wrap_content" android:textSize="25sp" android:id="@+id/btnExit"
android:layout_height="wrap_content" android:text="Return to home menu"
android:layout_below="@+id/btnSearch" android:layout_alignRight="@+id/btnSearch"
android:layout_marginTop="37dp" android:layout_alignLeft="@+id/btnDis"></Button>
            <Button android:layout_width="110dip" android:textSize="25sp" android:id="@+id/btnSearch"
android:layout_height="50dip" android:text="Search Equipment" android:layout_alignBottom="@+id/btnDis"
android:layout_alignLeft="@+id/tbtnSwitch"></Button>
            <ToggleButton android:textOff="Off" android:layout_width="110dip" android:layout_height="50dip"
android:textOn="On" android:id="@+id/tbtnSwitch" android:textSize="23sp" android:text="OFF"
android:layout_alignBottom="@+id/imageView1" android:layout_alignRight="@+id/imageView2"
android:layout_marginRight="52dp" android:layout_marginBottom="23dp"></ToggleButton>
            <ImageView android:layout_height="wrap_content" android:layout_width="wrap_content"
android:id="@+id/imageView1" android:src="@drawable/lanya" android:layout_above="@+id/btnSearch"
 android:layout_toLeftOf="@+id/tbtnSwitch" android:layout_marginRight="41dp"></ImageView>
            <ImageView android:layout_height="wrap_content" android:layout_width="wrap_content"
android:id="@+id/imageView2" android:src="@drawable/sousuo" android:layout_above="@+id/tbtnSwitch"
android:layout_centerHorizontal="true"></ImageView>
      </RelativeLayout>
</LinearLayout>
```

14.1.3 显示公司介绍信息

编写文件 company.xml，功能是显示公司介绍信息，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:background="@drawable/bac"
  android:layout_height="match_parent">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_width="match_parent"
 android:layout_height="match_parent">
        <ImageView android:src="@drawable/company" android:id="@+id/imageView1"
 android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_alignParentBottom="true" android:layout_centerHorizontal="true"></ImageView>
        <ImageButton android:src="@drawable/back" android:background="#0000"
android:layout_height="wrap_content" android:id="@+id/back" android:layout_width="wrap_content"
android:layout_alignParentBottom="true" android:layout_alignParentRight="true"></ImageButton>
        <ImageView android:src="@drawable/ctitle" android:id="@+id/imageView2"
 android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_alignParentTop="true" android:layout_alignRight="@+id/imageView1"
android:layout_marginRight="22dp"></ImageView>
    </RelativeLayout>
</LinearLayout>
```

14.1.4 系统功能介绍

编写文件 dialog.xml，功能是实现一个系统功能介绍对话框效果，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:id="@+id/dialog" android:weightSum="1">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_width="match_parent"
android:layout_height="wrap_content" android:layout_weight="1.09">
        <TextView android:text="@string/intr1" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView2"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView1"
android:layout_alignLeft="@+id/textView1" android:layout_marginLeft="24dp"></TextView>
        <TextView android:text="@string/intr0" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView1"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_alignParentTop="true"
android:layout_alignParentLeft="true" android:layout_marginLeft="50dp"
android:layout_marginTop="53dp"></TextView>
        <TextView android:text="@string/intr2" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView3"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView2"
android:layout_alignLeft="@+id/textView1"></TextView>
```

```
            <TextView android:text="@string/intr3" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView4"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView3"
android:layout_alignLeft="@+id/textView2"></TextView>
            <ImageView android:id="@+id/imageView1" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:src="@drawable/main_add_enable"
android:layout_below="@+id/textView4" android:layout_alignLeft="@+id/textView3"
android:layout_marginTop="37dp"></ImageView>
            <TextView android:text="@string/intr4" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView5"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_alignBottom="@+id/imageView1"
android:layout_toRightOf="@+id/imageView1" android:layout_marginLeft="34dp"></TextView>
            <ImageView android:id="@+id/imageView2" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:src="@drawable/main_menu_enable"
android:layout_below="@+id/imageView1" android:layout_alignLeft="@+id/imageView1"
android:layout_marginTop="24dp"></ImageView>
            <TextView android:text="@string/intr5" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView6"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_alignBottom="@+id/imageView2"
android:layout_alignLeft="@+id/textView5"></TextView>
            <ImageView android:id="@+id/imageView3" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:src="@drawable/main_back_enable"
android:layout_below="@+id/imageView2" android:layout_alignLeft="@+id/imageView2"
android:layout_marginTop="30dp"></ImageView>
            <TextView android:text="@string/intr6" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView7"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_alignBottom="@+id/imageView3"
android:layout_alignLeft="@+id/textView6"></TextView>
            <TextView android:text="@string/intr7" android:textColor="#00ff00"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/textView8"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/imageView3"
android:layout_alignLeft="@+id/imageView3" android:layout_marginTop="52dp"></TextView>
            <TextView android:text="@string/intr8" android:textColor="#00ff00"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/textView9"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView8"
android:layout_alignLeft="@+id/textView8" android:layout_marginTop="47dp"></TextView>
            <TextView android:text="@string/intr9" android:textColor="#00ff00"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/textView10"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView9"
android:layout_alignLeft="@+id/textView9" android:layout_marginTop="45dp"></TextView>
        </RelativeLayout>
</LinearLayout>
```

14.1.5　第一路调光设置界面

编写文件 first.xml，功能是通过单选按钮列表实现了第一路调光设置界面，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#0000"
```

```
    android:weightSum="1" android:orientation="vertical">
    <ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/help_2"
    android:background="#0000"
    android:id="@+id/imageButton1"
    android:layout_gravity="right"></ImageButton>


    <TextView
    android:id="@+id/textview1"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
    android:layout_marginRight="76dp"
    android:layout_marginTop="15dp"></TextView>
    <RelativeLayout
    android:id="@+id/relativeLayout1"
    android:layout_width="match_parent" android:layout_height="696dp" android:gravity="left">
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup1"
android:layout_width="wrap_content" android:layout_alignParentTop="true"
android:layout_toRightOf="@+id/imageButton2" android:layout_marginLeft="18dp"
android:layout_marginTop="32dp">
        </RadioGroup>
        <ImageButton android:id="@+id/imageButton5" android:layout_height="wrap_content"
android:background="#0000" android:layout_width="wrap_content" android:src="@drawable/button2_c"
android:layout_alignTop="@+id/imageButton3"
android:layout_alignLeft="@+id/imageButton4"></ImageButton>
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup3"
android:layout_width="wrap_content" android:layout_alignTop="@+id/radioGroup2"
android:layout_toRightOf="@+id/radioGroup1" android:layout_marginLeft="45dp">
            <RadioButton android:text="0%" android:layout_height="wrap_content"
android:id="@+id/radio21" android:layout_width="wrap_content" ></RadioButton>
            <RadioButton android:text="20%" android:layout_height="wrap_content"
android:id="@+id/radio22" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="40%" android:layout_height="wrap_content"
android:id="@+id/radio23" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="60%" android:layout_height="wrap_content"
android:id="@+id/radio24" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="80%" android:layout_height="wrap_content"
android:id="@+id/radio25" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="100%" android:layout_height="wrap_content"
android:id="@+id/radio26" android:layout_width="wrap_content"></RadioButton>
        </RadioGroup>
        <ImageButton android:id="@+id/imageButton4" android:layout_height="wrap_content"
android:background="#0000" android:layout_width="wrap_content" android:src="@drawable/button2_o"
android:layout_alignTop="@+id/imageButton2" android:layout_toRightOf="@+id/radioGroup1"
 android:layout_marginLeft="24dp"></ImageButton>
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup4"
```

```
android:layout_width="wrap_content" android:layout_alignTop="@+id/radioGroup3"
android:layout_toRightOf="@+id/imageButton4" android:layout_marginLeft="76dp">
            <RadioButton android:text="0%" android:layout_height="wrap_content"
android:id="@+id/radio31" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="20%" android:layout_height="wrap_content"
android:id="@+id/radio32" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="40%" android:layout_height="wrap_content"
android:id="@+id/radio33" android:layout_width="wrap_content"></RadioButton>
        <RadioButton android:text="60%" android:layout_width="wrap_content" android:id="@+id/radio34"
android:layout_height="wrap_content" android:layout_below="@+id/radioGroup4"
android:layout_alignLeft="@+id/radioGroup4"></RadioButton>
        <RadioButton android:text="80%" android:layout_width="wrap_content" android:id="@+id/radio35"
android:layout_height="wrap_content" android:layout_below="@+id/radioButton4"
 android:layout_alignLeft="@+id/radioButton4"></RadioButton>
        <RadioButton android:text="100%" android:layout_width="wrap_content" android:id="@+id/radio36"
android:layout_height="wrap_content" android:layout_below="@+id/radioButton5"
android:layout_alignLeft="@+id/radioButton5"></RadioButton>
</RadioGroup>
              <TextView android:layout_height="wrap_content" android:text="第三路调光"
android:textColor="#ffffff" android:id="@+id/textView4" android:layout_width="wrap_content"
android:layout_alignTop="@+id/textView2" android:layout_alignLeft="@+id/radioGroup4"></TextView>
        <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button3_c" android:background="#0000" android:id="@+id/imageButton7"
android:layout_alignTop="@+id/imageButton5"
android:layout_alignLeft="@+id/imageButton6"></ImageButton>
        <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button3_o" android:background="#0000" android:id="@+id/imageButton6"
android:layout_alignTop="@+id/imageButton4" android:layout_toRightOf="@+id/imageButton4"
android:layout_marginLeft="56dp"></ImageButton>
        <TextView android:layout_width="wrap_content" android:text="第四路调光"
android:layout_height="wrap_content" android:id="@+id/textView5" android:textColor="#ffffff"
android:layout_alignTop="@+id/textView4" android:layout_alignRight="@+id/radioGroup5"></TextView>
        <ImageButton android:src="@drawable/button4_c" android:layout_height="wrap_content"
android:id="@+id/imageButton9" android:background="#0000" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton7"
android:layout_alignLeft="@+id/imageButton8"></ImageButton>
        <ImageButton android:src="@drawable/split" android:layout_height="wrap_content"
android:id="@+id/imageButton10" android:background="#0000" android:layout_width="1000dip"
android:layout_alignTop="@+id/radioGroup2" android:layout_alignRight="@+id/radioGroup4"
android:layout_marginTop="47dp"></ImageButton>
        <ImageButton android:src="@drawable/split" android:layout_height="wrap_content"
android:id="@+id/imageButton11" android:background="#0000" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton10"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
        <ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton12" android:layout_width="wrap_content"
android:layout_below="@+id/imageButton10" android:layout_alignLeft="@+id/radioGroup2"
android:layout_marginTop="35dp"></ImageButton>
        <ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton15" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton12"
```

```
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
            <ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton13" android:layout_width="wrap_content"
android:layout_below="@+id/imageButton12" android:layout_alignRight="@+id/imageButton4"
android:layout_marginTop="43dp"></ImageButton>
            <ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton17" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton13"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
            <ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton18" android:layout_width="wrap_content"
android:layout_below="@+id/imageButton13" android:layout_alignLeft="@+id/imageButton13"
android:layout_marginTop="39dp"></ImageButton>
            <ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton19" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton18"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
            <ImageButton android:id="@+id/imageButton3" android:layout_width="wrap_content"
android:src="@drawable/button1_c" android:layout_height="wrap_content" android:background="#0000"
android:layout_below="@+id/imageButton2" android:layout_alignLeft="@+id/imageButton2"
android:layout_marginTop="30dp"></ImageButton>
            <RadioGroup android:layout_width="wrap_content" android:id="@+id/radioGroup2"
 android:layout_height="wrap_content" android:layout_below="@+id/radioGroup1"
android:layout_alignParentLeft="true" android:layout_marginLeft="179dp" android:layout_marginTop="105dp">
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="0%" android:id="@+id/radio11"></RadioButton>
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="20%" android:id="@+id/radio12"></RadioButton>
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="40%" android:id="@+id/radio13"></RadioButton>
                <RadioButton    android:layout_height="wrap_content"    android:layout_width="wrap_content"
android:text="60%" android:id="@+id/radio14"></RadioButton>
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="80%" android:id="@+id/radio15"></RadioButton>
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="100%" android:id="@+id/radio16"></RadioButton>
            </RadioGroup>
            <TextView android:text="第二路调光" android:textColor="#ffffff" android:layout_width="wrap_content"
android:id="@+id/textView2" android:layout_height="wrap_content" android:layout_alignTop="@+id/textView3"
android:layout_alignLeft="@+id/radioGroup3"></TextView>
            <TextView android:text="第一路调光" android:textColor="#ffffff" android:layout_width="wrap_content"
android:id="@+id/textView3" android:layout_height="wrap_content" android:layout_below="@+id/radioGroup1"
android:layout_alignLeft="@+id/radioGroup2" android:layout_marginTop="44dp"></TextView>
            <TextView android:text="    一" android:layout_width="wrap_content" android:id="@+id/textView7"
android:textSize="30sp" android:layout_height="wrap_content" android:layout_below="@+id/imageButton11"
android:layout_alignRight="@+id/textView6"></TextView>
            <TextView android:text="    调" android:layout_width="wrap_content" android:id="@+id/textView8"
android:textSize="30sp" android:layout_height="wrap_content" android:layout_below="@+id/imageButton15"
 android:layout_alignRight="@+id/textView7"></TextView>
            <TextView android:text="    光" android:layout_width="wrap_content" android:id="@+id/textView9"
android:textSize="30sp" android:layout_height="wrap_content" android:layout_below="@+id/imageButton17"
```

```
android:layout_alignRight="@+id/textView8"></TextView>
            <TextView android:text="        界" android:layout_width="wrap_content"
android:id="@+id/textView10" android:textSize="30sp" android:layout_height="wrap_content"
android:layout_below="@+id/imageButton19" android:layout_alignRight="@+id/textView9"></TextView>
            <ImageButton android:id="@+id/imageButton8" android:layout_width="wrap_content"
android:src="@drawable/button4_o" android:layout_height="wrap_content" android:background="#0000"
android:layout_alignTop="@+id/imageButton6" android:layout_toRightOf="@+id/imageButton6"
android:layout_marginLeft="39dp"></ImageButton>
            <RadioGroup android:layout_width="wrap_content" android:id="@+id/radioGroup5"
android:layout_height="wrap_content" android:layout_alignTop="@+id/radioGroup4"
android:layout_alignRight="@+id/imageButton8">
                  <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="0%" android:id="@+id/radio41"></RadioButton>
                  <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="20%" android:id="@+id/radio42"></RadioButton>
                  <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="40%" android:id="@+id/radio43"></RadioButton>
                  <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="60%" android:id="@+id/radio44"></RadioButton>
                  <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="80%" android:id="@+id/radio45"></RadioButton>
                  <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="100%" android:id="@+id/radio46"></RadioButton>
            </RadioGroup>
            <ImageButton android:id="@+id/imageButton14" android:layout_width="1000dip"
android:src="@drawable/line" android:layout_height="5dip"
android:layout_below="@+id/radioGroup3"></ImageButton>
            <ImageButton android:layout_width="5dip" android:src="@drawable/line"
 android:layout_height="1000dip" android:id="@+id/imageButton16" android:layout_alignParentTop="true"
android:layout_toRightOf="@+id/textView5" android:layout_marginLeft="46dp"></ImageButton>
            <TextView android:textSize="30sp" android:layout_width="wrap_content" android:text="        第"
android:id="@+id/textView6" android:layout_height="wrap_content"
android:layout_alignTop="@+id/radioGroup5" android:layout_toRightOf="@+id/imageButton16"
android:layout_marginLeft="24dp"></TextView>
            <ImageButton android:layout_width="wrap_content" android:src="@drawable/button1_add_x"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton2"
android:layout_below="@+id/imageButton14" android:layout_alignRight="@+id/radioGroup2"
android:layout_marginTop="56dp"></ImageButton>
            <TextView android:id="@+id/textView11" android:text="        面" android:textSize="30sp"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_above="@+id/imageButton14" android:layout_alignLeft="@+id/textView10"></TextView>
            <ImageButton android:layout_height="wrap_content" android:background="#0000"
android:id="@+id/btnopen" android:layout_width="wrap_content" android:src="@drawable/allop"
 android:layout_alignTop="@+id/imageButton8" android:layout_alignLeft="@+id/textView11"></ImageButton>
            <ImageButton android:layout_height="wrap_content" android:id="@+id/btnclose"
android:layout_width="wrap_content" android:src="@drawable/allcl" android:background="#0000"
android:layout_alignTop="@+id/imageButton9" android:layout_alignLeft="@+id/btnopen"></ImageButton>


    </RelativeLayout>

</LinearLayout>
```

14.1.6 执行主界面

编写文件 home.xml，此文件是系统执行后进入的主界面，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:background="@drawable/bac"
  android:layout_height="match_parent" android:weightSum="1">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_width="match_parent" android:layout_
height="682dp">
            <ImageView android:layout_alignParentLeft="true" android:id="@+id/imageView2" android:src=
"@drawable/enter1" android:layout_height="wrap_content" android:layout_width="wrap_content" android:layout
_below="@+id/imageView1"></ImageView>
            <ImageView android:id="@+id/imageView4" android:src="@drawable/enter2" android:layout_height=
"wrap_content" android:layout_width="wrap_content" android:layout_below="@+id/imageView2" android:
layout_alignParentLeft="true" android:layout_marginLeft="48dp" android:layout_marginTop="86dp"></ImageView>
            <ImageView android:id="@+id/imageView5" android:src="@drawable/enter3" android:layout_height=
"wrap_content" android:layout_width="wrap_content" android:layout_alignParentBottom="true" android:layout_
alignLeft= "@+id/imageView4" android:layout_marginBottom="66dp"></ImageView>
            <ImageView android:id="@+id/imageView7" android:src="@drawable/split" android:layout_height=
"wrap_content" android:layout_width="wrap_content" android:layout_alignBottom="@+id/imageView5" android:
 layout_alignLeft="@+id/imageView6"></ImageView>
            <ImageButton android:id="@+id/Enterp" android:background="#0000" android:src="@drawable/
enter0" android:layout_height="wrap_content" android:layout_width="wrap_content" android:layout_
alignBottom="@+id/imageView6" android:layout_alignLeft="@+id/Enterr"></ImageButton>
            <ImageButton android:id="@+id/Enterc" android:background="#0000" android:src="@drawable/
enter0" android:layout_height="wrap_content" android:layout_width="wrap_content" android:layout_above=
"@+id/imageView7" android:layout_alignLeft="@+id/Enterp"></ImageButton>
            <ImageView android:id="@+id/imageView6" android:src="@drawable/split" android:layout_height=
"wrap_content" android:layout_width="wrap_content" android:layout_below="@+id/imageView4" android:
layout_alignLeft="@+id/imageView3"></ImageView>
            <ImageView android:id="@+id/imageView1" android:src="@drawable/logo" android:layout_height=
"wrap_content" android:layout_width="wrap_content" android:layout_alignParentTop="true" android:layout_
centerHorizontal="true"></ImageView>
            <ImageButton android:background="#0000" android:id="@+id/Enterr" android:src="@drawable/enter0"
android:layout_height="wrap_content" android:layout_width="wrap_content" android:layout_alignBottom="@+id/
imageView3" android:layout_toRightOf="@+id/imageView1" android:layout_marginLeft="63dp"></ImageButton>
            <ImageView android:id="@+id/imageView3" android:src="@drawable/split" android:layout_height=
"wrap_content" android:layout_width="wrap_content" android:layout_below="@+id/imageView2" android:layout_
alignRight="@+id/imageView1" android:layout_marginRight="87dp"></ImageView>

</RelativeLayout>
    <RelativeLayout android:layout_weight="0.95" android:layout_height="wrap_content" android:id="@+id/
relativeLayout2" android:layout_width="match_parent"></RelativeLayout>
    <RelativeLayout android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:layout_alignParentBottom="true"
         style="@android:style/ButtonBar"
```

```
        android:background="@drawable/title_background">
            <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content" android:
background="#0000" android:id="@+id/back" android:src="@drawable/back" android:layout_centerHorizontal=
"true" android:layout_alignTop="@+id/imageButton3"></ImageButton>                    ></RelativeLayout>
</LinearLayout>
```

14.1.7　不同房间的照明亮度参考值

编写文件 lightstandard.xml，功能是显示不同房间的照明亮度参考值，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@drawable/bac"
  android:id="@+id/stand" android:weightSum="1">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_width="match_parent" android:layout_
height="wrap_content" android:layout_weight="1.09">
        <ImageView android:layout_height="wrap_content" android:src="@drawable/twitter1" android:id= "@+id/
imageView1" android:background="#0000" android:layout_width="wrap_content" android:layout_ alignParentTop=
"true" android:layout_alignRight="@+id/imageView2" android:layout_marginRight="119dp"></ImageView>
        <ImageView android:layout_height="wrap_content" android:src="@drawable/stand1" android:id=
"@+id/imageView2" android:layout_width="wrap_content" android:layout_below="@+id/imageView1" android:
layout_ alignParentRight="true" android:layout_marginRight="173dp"></ImageView>
    </RelativeLayout>
</LinearLayout>
```

14.1.8　产品的详细介绍

编写文件 product.xml，功能是显示本产品的详细介绍信息，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:background="@drawable/bac"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_height="match_parent" android:layout_
width="match_parent">
        <ImageButton android:src="@drawable/back" android:background="#0000" android:layout_height=
"wrap_content" android:id="@+id/back" android:layout_width="wrap_content" android:layout_alignParent
Bottom="true" android:layout_alignParentRight="true"></ImageButton>
        <ImageView android:src="@drawable/product" android:id="@+id/imageView1" android:layout_ height="
wrap_content" android:layout_width="wrap_content" android:layout_above="@+id/back" android:layout_ align
ParentLeft= "true" android:layout_marginLeft="117dp" android:layout_marginBottom="31dp"></ImageView>
        <ImageView android:src="@drawable/title" android:id="@+id/imageView2" android:layout_height=
"wrap_content" android:layout_width="wrap_content" android:layout_alignParentTop="true" android:layout_
alignLeft="@+id/imageView1" android:layout_marginLeft="59dp" android:layout_marginTop="42dp"></ImageView>
    </RelativeLayout>
</LinearLayout>
```

14.1.9 五路调光设置界面

编写文件 second.xml，功能是通过单选按钮列表实现一个五路调光设置界面效果，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#0000"
    android:weightSum="1" android:orientation="vertical">
    <ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/help_2"
    android:background="#0000"
    android:id="@+id/imageButton1"
    android:layout_gravity="right"></ImageButton>


    <TextView
    android:id="@+id/textview1"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
    android:layout_marginRight="76dp"
    android:layout_marginTop="15dp"></TextView>
    <RelativeLayout
    android:id="@+id/relativeLayout1"
    android:layout_width="match_parent" android:layout_height="696dp" android:gravity="left">
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup1"
android:layout_width="wrap_content" android:layout_alignParentTop="true"
android:layout_toRightOf="@+id/imageButton2" android:layout_marginLeft="18dp"
android:layout_marginTop="32dp">
        </RadioGroup>
        <ImageButton android:layout_width="wrap_content" android:id="@+id/imageButton3"
android:background="#0000" android:layout_height="wrap_content" android:src="@drawable/button5_c"
android:layout_below="@+id/imageButton2" android:layout_alignLeft="@+id/imageButton2"
android:layout_marginTop="26dp"></ImageButton>
        <ImageButton android:id="@+id/imageButton5" android:layout_height="wrap_content"
android:background="#0000" android:layout_width="wrap_content" android:src="@drawable/button6_c"
android:layout_alignTop="@+id/imageButton3"
android:layout_alignLeft="@+id/imageButton4"></ImageButton>
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup3"
android:layout_width="wrap_content" android:layout_alignTop="@+id/radioGroup2"
android:layout_toRightOf="@+id/radioGroup1" android:layout_marginLeft="45dp">
            <RadioButton android:text="0%" android:layout_height="wrap_content"
android:id="@+id/radio21" android:layout_width="wrap_content" ></RadioButton>
            <RadioButton android:text="20%" android:layout_height="wrap_content"
```



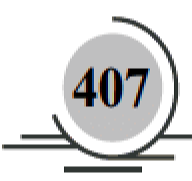
```
 android:id="@+id/radio22" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="40%" android:layout_height="wrap_content"
android:id="@+id/radio23" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="60%" android:layout_height="wrap_content"
android:id="@+id/radio24" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="80%" android:layout_height="wrap_content"
android:id="@+id/radio25" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="100%" android:layout_height="wrap_content"
android:id="@+id/radio26" android:layout_width="wrap_content"></RadioButton>
        </RadioGroup>
        <ImageButton android:id="@+id/imageButton4" android:layout_height="wrap_content"
android:background="#0000" android:layout_width="wrap_content" android:src="@drawable/button6_o"
android:layout_alignTop="@+id/imageButton2" android:layout_toRightOf="@+id/radioGroup1"
android:layout_marginLeft="24dp"></ImageButton>
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup4"
android:layout_width="wrap_content" android:layout_alignTop="@+id/radioGroup3"
android:layout_toRightOf="@+id/imageButton4" android:layout_marginLeft="76dp">
            <RadioButton android:text="0%" android:layout_height="wrap_content"
android:id="@+id/radio31" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="20%" android:layout_height="wrap_content"
android:id="@+id/radio32" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="40%" android:layout_height="wrap_content"
android:id="@+id/radio33" android:layout_width="wrap_content"></RadioButton>
         <RadioButton android:text="60%" android:layout_width="wrap_content" android:id="@+id/radio34"
android:layout_height="wrap_content" android:layout_below="@+id/radioGroup4"
android:layout_alignLeft="@+id/radioGroup4"></RadioButton>
         <RadioButton android:text="80%" android:layout_width="wrap_content" android:id="@+id/radio35"
android:layout_height="wrap_content" android:layout_below="@+id/radioButton4"
android:layout_alignLeft="@+id/radioButton4"></RadioButton>
         <RadioButton android:text="100%" android:layout_width="wrap_content" android:id="@+id/radio36"
android:layout_height="wrap_content" android:layout_below="@+id/radioButton5"
android:layout_alignLeft="@+id/radioButton5"></RadioButton></RadioGroup>
              <TextView android:layout_height="wrap_content" android:text="第七路调光"
android:textColor="#ffffff" android:id="@+id/textView4" android:layout_width="wrap_content"
 android:layout_alignTop="@+id/textView2" android:layout_alignLeft="@+id/radioGroup4"></TextView>
         <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button7_c" android:background="#0000" android:id="@+id/imageButton7"
android:layout_alignTop="@+id/imageButton5"
android:layout_alignLeft="@+id/imageButton6"></ImageButton>
         <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button7_o" android:background="#0000" android:id="@+id/imageButton6"
android:layout_alignTop="@+id/imageButton4" android:layout_toRightOf="@+id/imageButton4"
android:layout_marginLeft="56dp"></ImageButton>
         <TextView android:layout_height="wrap_content" android:text="第八路调光"
android:textColor="#ffffff" android:id="@+id/textView5" android:layout_width="wrap_content"
android:layout_alignTop="@+id/textView4" android:layout_alignLeft="@+id/radioGroup5"></TextView>
         <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button8_c" android:background="#0000" android:id="@+id/imageButton9"
android:layout_alignTop="@+id/imageButton7"
android:layout_alignLeft="@+id/imageButton8"></ImageButton>
         <RadioGroup android:id="@+id/radioGroup2" android:layout_width="wrap_content"
```

```
 android:layout_height="wrap_content" android:layout_below="@+id/radioGroup1"
android:layout_alignParentLeft="true" android:layout_marginLeft="207dp" android:layout_marginTop="84dp">
                <RadioButton android:text="0%" android:id="@+id/radio11"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="20%" android:id="@+id/radio12"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="40%" android:id="@+id/radio13"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="60%" android:id="@+id/radio14"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="80%" android:id="@+id/radio15"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="100%" android:id="@+id/radio16"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
           </RadioGroup>
           <TextView android:textColor="#ffffff" android:layout_width="wrap_content" android:text="第五路调光"
android:id="@+id/textView3" android:layout_height="wrap_content" android:layout_below="@+id/radioGroup1"
android:layout_alignLeft="@+id/radioGroup2" android:layout_marginTop="31dp"></TextView>
           <ImageButton android:layout_width="wrap_content" android:src="@drawable/button5_o"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton2"
 android:layout_below="@+id/radioGroup2" android:layout_alignRight="@+id/radioGroup2"
android:layout_marginTop="76dp"></ImageButton>
           <TextView android:textColor="#ffffff" android:layout_width="wrap_content" android:text="第六路调光"
android:id="@+id/textView2" android:layout_height="wrap_content" android:layout_alignTop="@+id/textView3"
android:layout_alignLeft="@+id/radioGroup3"></TextView>
           <RadioGroup android:id="@+id/radioGroup5" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignTop="@+id/radioGroup4"
android:layout_toRightOf="@+id/imageButton6" android:layout_marginLeft="59dp">
                <RadioButton android:text="0%" android:id="@+id/radio41"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="20%" android:id="@+id/radio42"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="40%" android:id="@+id/radio43"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="60%" android:id="@+id/radio44"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="80%" android:id="@+id/radio45"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
                <RadioButton android:text="100%" android:id="@+id/radio46"
 android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
           </RadioGroup>
           <ImageButton android:layout_width="wrap_content" android:src="@drawable/button8_o"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton8"
android:layout_alignTop="@+id/imageButton6" android:layout_toRightOf="@+id/imageButton6"
android:layout_marginLeft="39dp"></ImageButton>
           <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton10"
android:layout_alignTop="@+id/radioGroup3" android:layout_alignRight="@+id/radioGroup3"
android:layout_marginTop="42dp"></ImageButton>
           <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton11"
android:layout_below="@+id/imageButton10" android:layout_alignRight="@+id/radioGroup3"
android:layout_marginTop="43dp"></ImageButton>
```

```
            <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton12"
android:layout_below="@+id/imageButton11" android:layout_alignRight="@+id/textView2"
android:layout_marginTop="42dp"></ImageButton>
            <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton13"
android:layout_alignBottom="@+id/radioGroup3" android:layout_alignRight="@+id/radioGroup3"
android:layout_marginBottom="46dp"></ImageButton>
            <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton14"
android:layout_alignTop="@+id/imageButton10" android:layout_alignRight="@+id/textView5"></ImageButton>
            <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton15"
android:layout_alignTop="@+id/imageButton11"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
            <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton16"
android:layout_alignTop="@+id/imageButton12"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
            <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton17"
android:layout_alignTop="@+id/imageButton13"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
            <ImageButton android:layout_width="1000dip" android:src="@drawable/line"
android:layout_height="5dip" android:id="@+id/imageButton18" android:layout_below="@+id/radioGroup3"
android:layout_alignParentLeft="true" android:layout_marginTop="29dp"></ImageButton>
            <ImageButton android:layout_width="5dip" android:src="@drawable/line"
android:layout_height="800dip" android:id="@+id/imageButton19" android:layout_alignParentTop="true"
android:layout_above="@+id/imageButton9" android:layout_toRightOf="@+id/imageButton9"></ImageButton>
            <TextView android:layout_width="wrap_content" android:text="    二" android:textSize="30sp"
android:id="@+id/textView7" android:layout_height="wrap_content"
android:layout_below="@+id/imageButton14" android:layout_alignRight="@+id/textView6"></TextView>
            <TextView android:layout_width="wrap_content" android:text="    调" android:textSize="30sp"
android:id="@+id/textView8" android:layout_height="wrap_content"
android:layout_below="@+id/imageButton15" android:layout_alignRight="@+id/textView7"></TextView>
            <TextView android:layout_width="wrap_content" android:text="    光" android:textSize="30sp"
android:id="@+id/textView9" android:layout_height="wrap_content"
android:layout_below="@+id/imageButton16" android:layout_alignRight="@+id/textView8"></TextView>
            <TextView android:layout_height="wrap_content" android:text="    界" android:textSize="30sp"
android:id="@+id/textView10" android:layout_width="wrap_content"
android:layout_above="@+id/imageButton17" android:layout_alignRight="@+id/textView9"></TextView>
            <TextView android:layout_height="wrap_content" android:text="    面" android:textSize="30sp"
android:id="@+id/textView11" android:layout_width="wrap_content"
android:layout_below="@+id/imageButton17" android:layout_alignRight="@+id/textView10"></TextView>
            <TextView android:layout_height="wrap_content" android:text="    第" android:textSize="30sp"
android:id="@+id/textView6" android:layout_width="wrap_content" android:layout_above="@+id/textView7"
android:layout_toRightOf="@+id/imageButton19"></TextView>


    </RelativeLayout>

</LinearLayout>
```



14.2 实现程序文件

知识点讲解：光盘:视频\知识点\第 14 章\实现程序文件.avi

在本章 14.1 的内容中，已经介绍了布局文件的设计过程。在本节的内容中，将详细讲解本系统实例的具体 Activity 程序文件的实现过程。

14.2.1 主 Activity

编写文件 Main.java，功能是响应用户按键处理事件，根据用户触摸的选项来到对应的模式，通过动画效果过渡来到 HOME 界面。文件 Main.java 的具体实现代码如下所示。

```
public class Main extends ActivityGroup implements OnGestureListener,OnTouchListener {
    //声明 ViewFlipper 对象
    private ViewFlipper m_ViewFlipper;
    //声明 GestureDetector 对象
    private GestureDetector m_GestureDetector;
    //声明 LocalActivityManager 对象
    private LocalActivityManager m_ActivityManager;
    private static int FLING_MIN_DISTANCE = 100;
    private static int FLING_MIN_VELOCITY = 200;
    //定义自定义图片加文字按钮 ImageButton 对象
    //private ImageButton mButton1;

    //单选按键部分 1
    private String[ ] areas = new String[ ]{"一般模式", "会议模式", "视频模式","迎接模式", "返回"};
    private RadioOnClick radioOnClick = new RadioOnClick(4);
    @SuppressWarnings("unused")
    private ListView RadioListView;
   private ImageButton imagebtn,imagebutton;
   OutputStream tmpOut = null;
   Timer timer=new Timer();
   @Override
   public void onCreate(Bundle savedInstanceState)
   {
       super.onCreate(savedInstanceState);
       requestWindowFeature(Window.FEATURE_NO_TITLE);
       //设置内容视图
       setContentView(R.layout.main);
       getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
       imagebutton=(ImageButton)findViewById(R.id.imageButton1);
       imagebutton.setOnClickListener(new ImageButton.OnClickListener()
       {
        @Override
           public void onClick(View v)
        {
```

```
                /*Intent intent=new Intent();
                intent.setClass(Main.this, stand.class);
                startActivity(intent);
                Main.this.finish();
                overridePendingTransition(R.anim.zoomin,R.anim.zoomout);*/
                LayoutInflater inflater = getLayoutInflater();
                  View layout = inflater.inflate(R.layout.lightstandard,
                  (ViewGroup) findViewById(R.id.stand));

                   new AlertDialog.Builder(Main.this)
                   .setTitle("Ihe nationl atandard of illumination")
                   .setView(layout)
                   .setPositiveButton("Return", null)
                   //.setNegativeButton("取消", null)
                   .show();
          }
        }
                );
        //单选按键部分 2
        imagebtn=(ImageButton)findViewById(R.id.imageButton2);
        imagebtn.setOnClickListener(new RadioClickListener());
        //构建 ViewFlipper 对象
     m_ViewFlipper = (ViewFlipper) findViewById(R.id.fliper);
     //获取 Activity 消息
     m_ActivityManager = getLocalActivityManager();
     //注册一个用于手势识别的类
     m_GestureDetector = new GestureDetector(this);
     //添加视图，指定每个视图对应的 Activity
          m_ViewFlipper.addView((m_ActivityManager.startActivity("", new Intent(Main.this,Firstpage.class)).
getDecorView()),0);
     m_ViewFlipper.addView((m_ActivityManager.startActivity("", new Intent(Main.this,Secondpage.class)).
getDecorView()),1);
     //给 ViewFlipper 设置一个 listener
     m_ViewFlipper.setOnTouchListener(this);
     //默认为正在播放页面并设置图标
     //设置相应元素索引显示的子视图
          m_ViewFlipper.setDisplayedChild(0);
     //允许长按住 ViewFlipper，这样才能识别拖动等手势
     m_ViewFlipper.setLongClickable(true);
     //监听
         /** Called when the activity is first created. */
        /*取得 Button 对象*/
     //返回按钮按键事件
       /* btnbac = (ImageButton) findViewById(R.id.btnback);
        btnbac.setOnClickListener(new ImageButton.OnClickListener(){
        @Override
        public void onClick(View v)
        {
           // TODO Auto-generated method stub
          Intent intent=new Intent();
          intent.setClass(Main.this, home.class);
```

```
            startActivity(intent);
            Main.this.finish();
            overridePendingTransition(R.anim.zoomin,R.anim.zoomout);
            /*Intent intent = new Intent();
            intent.setClass(Main.this, home.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);    //注意本行的 FLAG 设置
            startActivity(intent);

        } });*/}
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
            //按下键盘上的返回键
            if(keyCode == KeyEvent.KEYCODE_BACK){
                    new AlertDialog.Builder(this)
                            //.setIcon(R.drawable.services)
                   .setTitle(R.string.app_about)
                     /*设置弹出窗口的样式*/
                   // .setIcon(R.drawable.hot)
                     /*设置弹出窗口的信息*/
                     .setMessage(R.string.app_about_msg)
                     .setPositiveButton(R.string.str_ok,
                     new DialogInterface.OnClickListener()
                  {
                   public void onClick(DialogInterface dialoginterface, int i)
                   {
                     finish();/*关闭窗口*/
                   }
                   }
                  )
                     /*设置弹出窗口的返回事件*/
                   .setNegativeButton(R.string.str_no,
                       new DialogInterface.OnClickListener()
                     {
                   public void onClick(DialogInterface dialoginterface, int i)
                  {
                  }
                     })
                  .show();
                    //setResult(11,this.getIntent());
                    return true;
            }else{
                    return super.onKeyDown(keyCode, event);
            }
    }
    @Override
    protected void onDestroy() {
            super.onDestroy();


            android.os.Process.killProcess(android.os.Process.myPid());
            System.exit(0);
```

```
        overridePendingTransition(R.anim.zoomin,R.anim.zoomout);
        //或者下面这种方式




 }
 /**
  * 定义从右侧进入的动画效果
  * @return
  */
public Animation inFromRightAnimation()
{
 Animation inFromRight = new TranslateAnimation(
              Animation.RELATIVE_TO_PARENT, +1.0f,
              Animation.RELATIVE_TO_PARENT, 0.0f,
              Animation.RELATIVE_TO_PARENT, 0.0f,
              Animation.RELATIVE_TO_PARENT, 0.0f);
 inFromRight.setDuration(500);
 inFromRight.setInterpolator(new AccelerateInterpolator());
 return inFromRight;
}
/**
  * 定义从左侧退出的动画效果
  * @return
  */
public Animation outToLeftAnimation()
{
 Animation outtoLeft = new TranslateAnimation(
              Animation.RELATIVE_TO_PARENT, 0.0f,
              Animation.RELATIVE_TO_PARENT, -1.0f,
              Animation.RELATIVE_TO_PARENT, 0.0f,
              Animation.RELATIVE_TO_PARENT, 0.0f);
 outtoLeft.setDuration(500);
 outtoLeft.setInterpolator(new AccelerateInterpolator());
 return outtoLeft;
}
/**
  * 定义从左侧进入的动画效果
  * @return
  */
public Animation inFromLeftAnimation()
{
 Animation inFromLeft = new TranslateAnimation(
              Animation.RELATIVE_TO_PARENT, -1.0f,
              Animation.RELATIVE_TO_PARENT, 0.0f,
              Animation.RELATIVE_TO_PARENT, 0.0f,
              Animation.RELATIVE_TO_PARENT, 0.0f);
 inFromLeft.setDuration(500);
 inFromLeft.setInterpolator(new AccelerateInterpolator());
 return inFromLeft;
}
```

```
/**
 * 定义从右侧退出时的动画效果
 * @return
 */
public Animation outToRightAnimation()
{
 Animation outtoRight = new TranslateAnimation(
         Animation.RELATIVE_TO_PARENT, 0.0f,
         Animation.RELATIVE_TO_PARENT, +1.0f,
         Animation.RELATIVE_TO_PARENT, 0.0f,
         Animation.RELATIVE_TO_PARENT, 0.0f);
 outtoRight.setDuration(500);
 outtoRight.setInterpolator(new AccelerateInterpolator());
 return outtoRight;
}

 @Override
 public boolean onDown(MotionEvent e) {
     return false;
 }
 @Override
 public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
         float velocityY) {
     //当向左侧滑动的时候
 if(e1.getX()-e2.getX()>FLING_MIN_DISTANCE && Math.abs(velocityX)>FLING_MIN_VELOCITY)
 {
     //设置 View 进入屏幕时使用的动画
     m_ViewFlipper.setInAnimation(inFromRightAnimation());
     //设置 View 退出屏幕时使用的动画
     m_ViewFlipper.setOutAnimation(outToLeftAnimation());
     //下一个页面
     m_ViewFlipper.showNext();
     //获取相应元素索引显示的子视图
 }
 //当向右侧滑动的时候
     else if(e2.getX()-e1.getX()>FLING_MIN_DISTANCE && Math.abs(velocityX)>FLING_MIN_VELOCITY)
     {
         //设置 View 进入屏幕时使用的动画
         m_ViewFlipper.setInAnimation(inFromLeftAnimation());
         //设置 View 退出屏幕时使用的动画
         m_ViewFlipper.setOutAnimation(outToRightAnimation());
         //上一个页面
         m_ViewFlipper.showPrevious();
         //获取相应元素索引显示的子视图
     }
     return false;
 }
 @Override
 public void onLongPress(MotionEvent e) {

 }
```

```
    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,
            float distanceY) {

        return false;
    }
    @Override
    public void onShowPress(MotionEvent e) {

    }
    @Override
    public boolean onSingleTapUp(MotionEvent e) {
        return false;
    }
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        //一定要将触屏事件交给手势识别类去处理(自己处理会很麻烦的)
        return m_GestureDetector.onTouchEvent(event);
    }

   //单选按键部分 3
   class RadioClickListener implements OnClickListener {
      @Override
      public void onClick(View v) {
       AlertDialog ad =new AlertDialog.Builder(Main.this).setTitle("选择模式")
       .setSingleChoiceItems(areas,radioOnClick.getIndex(),radioOnClick).create();
       RadioListView=ad.getListView();
       ad.show();
   }
   }

   /**
    * 单击单选按钮事件
    *
    */
   class RadioOnClick implements DialogInterface.OnClickListener{
private int index;

public RadioOnClick(int index){
 this.index = index;
}
public void setIndex(int index){
 this.index=index;
}
public int getIndex(){
 return index;
}

public void onClick(DialogInterface dialog, int whichButton){
   setIndex(whichButton);
 // Toast.makeText(Main.this, "您选择了：  " + areas[index], Toast.LENGTH_LONG).show();
```

```
switch (index)
{
//一般模式下
    case    0:

    {
        //第一个子节点的亮度
        send(0x01,0x5a);

        //第二个子节点的亮度
        send(0x02,0x5a);

        //第三个子节点的亮度
        send(0x03,0x5a);

        //第四个子节点的亮度
        send(0x04,0x5a);
 Toast.makeText(Main.this, "您选择了：  " + areas[index], Toast.LENGTH_LONG).show();
    dialog.dismiss();
}
break;


// 会议模式下
    case    1:
    {
     //第一路子节点
     send(0x01,0x23);

     //第二路子节点
     send(0x02,0x23);

     //第三路子节点
     send(0x03,0x23);

     //第四路子节点
     send(0x04,0x23);
     Toast.makeText(Main.this, "您选择了：  " + areas[index], Toast.LENGTH_LONG).show();
     dialog.dismiss();
     }     break;


//视频模式下
    case    2:
    {
     //第一路子节点
     send(0x01,0xc8);

     //第二路子节点
     send(0x02,0xc8);
```

```
        //第三路子节点
        send(0x03,0xc8);

        //第四路子节点
        send(0x04,0xc8);

        Toast.makeText(Main.this, "您选择了：  " + areas[index], Toast.LENGTH_LONG).show();
        dialog.dismiss();
        }     break;


    //迎接模式下
       case   3:
       {
        //第一路子节点
        send(0x01,0x23);

        //第二路子节点
        TimerTask timerTask = new TimerTask() {
                 @Override
                 public void run()
                 {
                            //你要干的活
                    send(0x02,0x23);
                 }
       };
       timer.schedule(timerTask, 1000 * 3);
        //第三路子节点
        TimerTask timerTask1 = new TimerTask() {
             @Override
             public void run() {

                       //你要干的活
               send(0x03,0x23);

             }
       };
             timer.schedule(timerTask1, 1000 * 6); //2 秒后执行

        //第四路子节点
         TimerTask timerTask2 = new TimerTask() {
               @Override
               public void run() {
                              //你要干的活
                 send(0x04,0x23);

               }};
               timer.schedule(timerTask2, 1000 * 9);
        Toast.makeText(Main.this, "您选择了：  " + areas[index], Toast.LENGTH_LONG).show();
        dialog.dismiss();
        }     break;
```

```
    //返回
       case 4:

       {//Toast.makeText(Main.this, "您选择了：  " + areas[index], Toast.LENGTH_LONG).show();
         dialog.dismiss();   }
 }


}

  private void   send(int Room,int Grade){

            try {

                    //String strpass="@@UP....";
                   byte[ ] byteone=new byte[8];//=strpass.getBytes("US-ASCII");
                        byteone[0]=(byte)0xf5;
                        byteone[1]=(byte)0x5f;
                        byteone[2]=(byte)0x00;
                        byteone[3]=(byte)Room;
                        byteone[4]=(byte)0x03;
                        byteone[5]=(byte)0x00;
                        byteone[6]=(byte)Grade;
                        byteone[7]=(byte)0x06;

                      //byte[ ] bytekai=strpass.getBytes("US-ASCII");
               tmpOut = BluetoothMain.btSocket.getOutputStream();
               tmpOut.write(byteone);}
           catch (IOException e) {
               Log.e("BluetoothReadService", "temp sockets not created", e);
          }
             }
  }}
```

14.2.2 监听单击事件

编写文件 home.java，功能是监听用户触摸单击的选项来执行对应的处理程序，来到对应的 Activity 界面。文件 home.java 的具体实现代码如下所示。

```
public class home extends Activity {
      private ImageButton enterr,enterp,enterc,back;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         requestWindowFeature(Window.FEATURE_NO_TITLE);
         setContentView(R.layout.home);
         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
         enterr=(ImageButton) findViewById (R.id.Enterr);
```

```
enterp=(ImageButton) findViewById (R.id.Enterp);
enterc=(ImageButton) findViewById (R.id.Enterc);
back=(ImageButton) findViewById (R.id.back);
enterr.setOnClickListener(new Button.OnClickListener()
{

        @Override
        public void onClick(View v) {
              // TODO Auto-generated method stub
              Intent intent = new Intent();
        intent.setClass(home.this, BluetoothMain.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);   //注意本行的 FLAG 设置
        startActivity(intent);
         //home.this.finish();
        overridePendingTransition(R.anim.zoomin,R.anim.zoomout);


        }

}
        );

enterp.setOnClickListener(new Button.OnClickListener()
{
        @Override
        public void onClick(View v) {
              // TODO Auto-generated method stub
              Intent intent=new Intent();
        intent.setClass(home.this, product.class);
        startActivity(intent);
        home.this.finish();
        overridePendingTransition(R.anim.zoomin,R.anim.zoomout);

}}
        );

enterc.setOnClickListener(new Button.OnClickListener()

{

        @Override
        public void onClick(View v) {
              // TODO Auto-generated method stub
              Intent intent=new Intent();
        intent.setClass(home.this, company.class);
        startActivity(intent);
        home.this.finish();
        overridePendingTransition(R.anim.zoomin,R.anim.zoomout);

}}
        );
```

```
        back.setOnClickListener(new Button.OnClickListener()
         {

                @Override
                public void onClick(View v) {

                      AlertDialog.Builder alertbBuilder=new AlertDialog.Builder(home.this);
                          //.setIcon(R.drawable.services)
                           alertbBuilder.setTitle(R.string.app_about)
                      /*设置弹出窗口的图式*/
                      //.setIcon(R.drawable.hot)
                      /*设置弹出窗口的信息*/
                      .setMessage(R.string.app_about_msg)
                      .setPositiveButton(R.string.str_ok,
                      new DialogInterface.OnClickListener()
                    {
                     public void onClick(DialogInterface dialoginterface, int i)
                     {
                      finish();/*关闭窗口*/
                     }
                    }
                    )
                      /*设置弹出窗口的返回事件*/
                      .setNegativeButton(R.string.str_no,
                       new DialogInterface.OnClickListener()
                      {
                     public void onClick(DialogInterface dialoginterface, int i)
                    {
                    }
                      })
                    .show();

                }
    });}
}
```

14.2.3 设置系统的蓝牙参数

编写文件 BluetoothMain.java，功能是根据用户的设置选项来设置系统的蓝牙参数，实现控制本系统蓝牙设备的功能。文件 BluetoothMain.java 的具体实现代码如下所示。

```
public class BluetoothMain extends Activity {
     static final String SPP_UUID = "00001101-0000-1000-8000-00805F9B34FB";

     Button btnSearch, btnDis, btnExit;
     ToggleButton tbtnSwitch;
     ListView lvBTDevices;
     ArrayAdapter<String> adtDevices;
     List<String> lstDevices = new ArrayList<String>();
```

```
BluetoothAdapter btAdapt;
public static BluetoothSocket btSocket;

@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.bluetooth);
        // Button 设置
        btnSearch = (Button) this.findViewById(R.id.btnSearch);
        btnSearch.setOnClickListener(new ClickEvent());
        btnExit = (Button) this.findViewById(R.id.btnExit);
        btnExit.setOnClickListener(new ClickEvent());
        btnDis = (Button) this.findViewById(R.id.btnDis);
        btnDis.setOnClickListener(new ClickEvent());

        // ToogleButton 设置
        tbtnSwitch = (ToggleButton) this.findViewById(R.id.tbtnSwitch);
        tbtnSwitch.setOnClickListener(new ClickEvent());

        // ListView 及其数据源适配器
        lvBTDevices = (ListView) this.findViewById(R.id.lvDevices);
        adtDevices = new ArrayAdapter<String>(BluetoothMain.this,
                        android.R.layout.simple_list_item_1, lstDevices);
        lvBTDevices.setAdapter(adtDevices);
        //lvBTDevices.setOnItemClickListener(new ItemClickEvent());

        //初始化本机蓝牙功能，读取蓝牙状态并显示
        btAdapt = BluetoothAdapter.getDefaultAdapter();
      if (btAdapt.getState() == BluetoothAdapter.STATE_OFF)
                tbtnSwitch.setChecked(false);
        else if (btAdapt.getState() == BluetoothAdapter.STATE_ON)
                tbtnSwitch.setChecked(true);

        //注册 Receiver 来获取蓝牙设备相关的结果
        IntentFilter intent = new IntentFilter();
        intent.addAction(BluetoothDevice.ACTION_FOUND);// 用 BroadcastReceiver 来取得搜索结果
        intent.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
        intent.addAction(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
        intent.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
        registerReceiver(searchDevices, intent);

        if (btAdapt.getState() == BluetoothAdapter.STATE_OFF) {// 如果蓝牙还没开启
                Toast.makeText(BluetoothMain.this, "Bluetooth is openning，Just a minute , please", 1000).show();
                btAdapt.enable();
                tbtnSwitch.setChecked(true);
        }
        setTitle("The Bluetooth address：" + btAdapt.getAddress());
        lstDevices.clear();
        btAdapt.startDiscovery();

}
```

```
private BroadcastReceiver searchDevices = new BroadcastReceiver() {

    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        Bundle b = intent.getExtras();
        Object[ ] lstName = b.keySet().toArray();

        //显示所有收到的消息及其细节
        for (int i = 0; i < lstName.length; i++) {
            String keyName = lstName[i].toString();
            Log.e(keyName, String.valueOf(b.get(keyName)));
        }

        //搜索设备时，取得设备的 MAC 地址
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device = intent
                    .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            String str= device.getName() + "|" + device.getAddress();
            String str1 = device.getAddress();
        //if(str1.equals("00:11:08:01:06:78"))   //用来判断是否为我们所需要的蓝牙
        //{

            if (lstDevices.indexOf(str) == -1)//防止重复添加
                lstDevices.add(str); //获取设备名称和 mac 地址
            adtDevices.notifyDataSetChanged();
            /*  添加判断，如果为已配对或者已存 mac 地址的设备
             *  自动进行连接，并跳转到另一个 Activity
             * */
            if(true){    //未添加条件
                btAdapt.cancelDiscovery();
                //String str = lstDevices.get();
                //String[ ] values = str.split("\\|");
                //String address=values[1];
                //Log.e("address",values[1]);
                UUID uuid = UUID.fromString(SPP_UUID);
                BluetoothDevice btDev = btAdapt.getRemoteDevice(device.getAddress());
                try {
                    btSocket = btDev
                            .createRfcommSocketToServiceRecord(uuid);
                    btSocket.connect();

                    Intent intent1 = new Intent();
                    intent1.setClass(BluetoothMain.this, Main.class);
                    startActivity(intent1);
                    overridePendingTransition(R.anim.zoomin,R.anim.zoomout);
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
```

```
            //}
            }
        }
    };

    @Override
    protected void onDestroy() {
        this.unregisterReceiver(searchDevices);
        super.onDestroy();
        android.os.Process.killProcess(android.os.Process.myPid());
    }
    class ClickEvent implements View.OnClickListener {
        @Override
        public void onClick(View v) {
            if (v == btnSearch)//搜索蓝牙设备，在 BroadcastReceiver 显示结果
            {

                if (btAdapt.getState() == BluetoothAdapter.STATE_OFF) {//如果蓝牙还没开启
                    Toast.makeText(BluetoothMain.this, "请先打开蓝牙", 1000).show();
                    return;
                }

                setTitle("本机蓝牙地址：" + btAdapt.getAddress());
                lstDevices.clear();
                btAdapt.startDiscovery();
            } else if (v == tbtnSwitch) {//本机蓝牙启动/关闭
                if (btAdapt.getState() == BluetoothAdapter.STATE_OFF){
                    btAdapt.enable();
                    tbtnSwitch.setChecked(true);
                }

                else if (btAdapt.getState() == BluetoothAdapter.STATE_ON){
                    btAdapt.disable();
                    tbtnSwitch.setChecked(false);
                }
            } else if (v == btnDis)//本机可以被搜索
            {
                Intent discoverableIntent = new Intent(
                        BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
                discoverableIntent.putExtra(
                        BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);
                startActivity(discoverableIntent);
            } else if (v == btnExit) {
                try {
                    if (btSocket != null)
                        btSocket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
             if (btAdapt.getState() == BluetoothAdapter.STATE_ON){
```

```
                    btAdapt.disable();
                }
                    BluetoothMain.this.finish();
            }
        }
    }
```

14.2.4 控制第一路光线的亮度

编写文件 Firstpage.java，功能是监听用户在第一路调光单选按钮列表中的选择值，根据这个值来控制光线的亮度。文件 Firstpage.java 的具体实现代码如下所示。

```
public class Firstpage extends Activity {
    private ImageButton imagebutton;
    private ImageButton ibutton1_o,ibutton1_c,ibutton2_o,ibutton2_c,ibutton3_o,
    ibutton3_c,ibutton4_o,ibutton4_c,btnallop,btnallcl;
    RadioGroup radiogroup0,radiogroup1,radiogroup2,radiogroup3;
    RadioButton radio1,radio2,radio3,radio4,radio5,radio6;
    RadioButton radio11,radio12,radio13,radio14,radio15,radio16;
    RadioButton radio21,radio22,radio23,radio24,radio25,radio26;
    RadioButton radio31,radio32,radio33,radio34,radio35,radio36;
    OutputStream tmpOut = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.first);
        //全开按钮
        btnallop=(ImageButton) findViewById(R.id.btnopen);
        btnallop.setOnClickListener(new Button.OnClickListener(){
            @Override
            public void onClick(View v)
            {
             //第一路灯亮
             send(0x01,0x23);


             //第二路灯亮
             send(0x02,0x23);

             //第三路灯亮
             send(0x03,0x23);

             //第四路灯亮
             send(0x04,0x23);


            }});

        //全关按钮

        btnallcl=(ImageButton) findViewById(R.id.btnclose);
```

```
        btnallcl.setOnClickListener(new Button.OnClickListener(){
            @Override
            public void onClick(View v)
            {
             //第一路灯亮
             send(0x01,0xff);

             //第二路灯亮
             send(0x02,0xff);

             send(0x03,0xff);

             send(0x04,0xff);

            }
            });
        //产品介绍按钮
        imagebutton=(ImageButton) findViewById(R.id.imageButton1);
        imagebutton.setOnClickListener(new Button.OnClickListener(){
            @Override
            public void onClick(View v)
            {
              LayoutInflater inflater = getLayoutInflater();
             View layout = inflater.inflate(R.layout.dialog,
             (ViewGroup) findViewById(R.id.dialog));

              new AlertDialog.Builder(Firstpage.this)
              .setTitle("Introduce")
              .setView(layout)
              .setPositiveButton("Return", null)
              //.setNegativeButton("取消", null)
              .show();
            } });

    //第一子节点的占空比
        radiogroup0=(RadioGroup)findViewById(R.id.radioGroup2);
       radio1=(RadioButton)findViewById(R.id.radio11);
       radio2=(RadioButton)findViewById(R.id.radio12);
       radio3=(RadioButton)findViewById(R.id.radio13);
       radio4=(RadioButton)findViewById(R.id.radio14);
       radio5=(RadioButton)findViewById(R.id.radio15);
       radio6=(RadioButton)findViewById(R.id.radio16);
       radiogroup0.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

           @Override
           public void onCheckedChanged(RadioGroup group, int checkedId) {
               // TODO Auto-generated method stub

             switch (checkedId)
             {
             case R.id.radio11:
```

```
/*DisplayToast("灯光亮度 0%！");*/
Toast.makeText(Firstpage.this, "第一路灯光亮度为 0%！",
        Toast.LENGTH_SHORT).show();
{
    send(0x01,0xff);

}
    break;
case R.id.radio12:
    // DisplayToast("灯光亮度 20%！");
    Toast.makeText(Firstpage.this, "第一路灯光亮度为 20%！",
            Toast.LENGTH_SHORT).show();
{
    send(0x01,0xc8);

}
    break;
case R.id.radio13:
    //DisplayToast("灯光亮度 40%！");
    Toast.makeText(Firstpage.this, "第一路灯光亮度为 40%！",
            Toast.LENGTH_SHORT).show();
{
    send(0x01,0x91);

}
    break;
case R.id.radio14:
    //DisplayToast("灯光亮度 60%！");
    Toast.makeText(Firstpage.this, "第一路灯光亮度为 60%！",
            Toast.LENGTH_SHORT).show();
{
    send(0x01,0x5a);

}
    break;
case R.id.radio15:
    // DisplayToast("灯光亮度 80%！");
    Toast.makeText(Firstpage.this, "第一路灯光亮度为 80%！",
            Toast.LENGTH_SHORT).show();
{
    send(0x01,0x23);

}
    break;
default:
    // DisplayToast("灯光亮度 100%！");
    Toast.makeText(Firstpage.this, "第一路灯光亮度为 100%！",
            Toast.LENGTH_SHORT).show();
{
    send(0x01,0x00);
```

```
            }
                break;
            }

        }
    });
//第二子节点的占空比
    radiogroup1=(RadioGroup)findViewById(R.id.radioGroup3);
    radio11=(RadioButton)findViewById(R.id.radio21);
    radio12=(RadioButton)findViewById(R.id.radio22);
    radio13=(RadioButton)findViewById(R.id.radio23);
    radio14=(RadioButton)findViewById(R.id.radio24);
    radio15=(RadioButton)findViewById(R.id.radio25);
    radio16=(RadioButton)findViewById(R.id.radio26);
    radiogroup1.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            // TODO Auto-generated method stub

            switch (checkedId)
            {
            case R.id.radio21:
                    /*DisplayToast("灯光亮度 0%！");*/
                Toast.makeText(Firstpage.this, "第二路灯光亮度为 0%！",
                        Toast.LENGTH_SHORT).show();
            {
                send(0x02,0xff);

            }
                break;
            case R.id.radio22:
                  // DisplayToast("灯光亮度 20%！");
                Toast.makeText(Firstpage.this, "第二路灯光亮度为 20%！",
                            Toast.LENGTH_SHORT).show();
            {
                send(0x02,0xc8);

            }
                break;
            case R.id.radio23:
                  //DisplayToast("灯光亮度 40%！");
                Toast.makeText(Firstpage.this, "第二路灯光亮度为 40%！",
                            Toast.LENGTH_SHORT).show();
            {
                send(0x02,0x91);

            }
                break;
            case R.id.radio24:
                    //DisplayToast("灯光亮度 60%！");
```

```
                    Toast.makeText(Firstpage.this, "第二路灯光亮度为 60%！",
                                    Toast.LENGTH_SHORT).show();
                {
                    send(0x02,0x5a);

                }
                    break;
                case R.id.radio25:
                     // DisplayToast("灯光亮度 80%！");
                    Toast.makeText(Firstpage.this, "第二路灯光亮度为 80%！",
                                    Toast.LENGTH_SHORT).show();
                {
                    send(0x02,0x23);

                }
                    break;
                default:
                     // DisplayToast("灯光亮度 100%！");
                    Toast.makeText(Firstpage.this, "第二路灯光亮度为 100%！",
                                    Toast.LENGTH_SHORT).show();
                {
                    send(0x02,0x00);

                }
                    break;
                }

            }
        });

    //第三子节点的占空比
        radiogroup2=(RadioGroup)findViewById(R.id.radioGroup4);
        radio21=(RadioButton)findViewById(R.id.radio31);
        radio22=(RadioButton)findViewById(R.id.radio32);
        radio23=(RadioButton)findViewById(R.id.radio33);
        radio24=(RadioButton)findViewById(R.id.radio34);
        radio25=(RadioButton)findViewById(R.id.radio35);
        radio26=(RadioButton)findViewById(R.id.radio36);
        radiogroup2.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                // TODO Auto-generated method stub

                switch (checkedId)
                {
                case R.id.radio31:
                      /*DisplayToast("灯光亮度 0%！");*/
                    Toast.makeText(Firstpage.this, "第三路灯光亮度为 0%！",
                                Toast.LENGTH_SHORT).show();
                {
```

```
        send(0x03,0xff);

    }
        break;
    case R.id.radio32:
        // DisplayToast("灯光亮度 20%！");
        Toast.makeText(Firstpage.this, "第三路灯光亮度为 20%！",
                        Toast.LENGTH_SHORT).show();
    {
        send(0x03,0xc8);

    }
        break;
    case R.id.radio33:
        //DisplayToast("灯光亮度 40%！");
        Toast.makeText(Firstpage.this, "第三路灯光亮度为 40%！",
                        Toast.LENGTH_SHORT).show();
    {
        send(0x03,0x91);

    }
        break;
    case R.id.radio34:
         //DisplayToast("灯光亮度 60%！");
        Toast.makeText(Firstpage.this, "第三路灯光亮度为 60%！",
                        Toast.LENGTH_SHORT).show();
    {
        send(0x03,0x5a);

    }
        break;
    case R.id.radio35:
        // DisplayToast("灯光亮度 80%！");
        Toast.makeText(Firstpage.this, "第三路灯光亮度为 80%！",
                        Toast.LENGTH_SHORT).show();
    {
        send(0x03,0x23);

    }
        break;
    default:
        // DisplayToast("灯光亮度 100%！");
        Toast.makeText(Firstpage.this, "第三路灯光亮度为 100%！",
                        Toast.LENGTH_SHORT).show();
    {
        send(0x03,0x00);

    }
        break;
    }
```

```
            }
        });    //第四路

    //第四子节点的占空比
        radiogroup3=(RadioGroup)findViewById(R.id.radioGroup5);
        radio31=(RadioButton)findViewById(R.id.radio41);
        radio32=(RadioButton)findViewById(R.id.radio42);
        radio33=(RadioButton)findViewById(R.id.radio43);
        radio34=(RadioButton)findViewById(R.id.radio44);
        radio35=(RadioButton)findViewById(R.id.radio45);
        radio36=(RadioButton)findViewById(R.id.radio46);
        radiogroup3.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                // TODO Auto-generated method stub

              switch (checkedId)
              {
              case R.id.radio41:
                    /*DisplayToast("灯光亮度 0%！");*/
                  Toast.makeText(Firstpage.this, "第四路灯光亮度为 0%！",
                            Toast.LENGTH_SHORT).show();
              {
                  send(0x04,0xff);

              }
                  break;
              case R.id.radio42:
                   // DisplayToast("灯光亮度 20%！");
                  Toast.makeText(Firstpage.this, "第四路灯光亮度为 20%！",
                                Toast.LENGTH_SHORT).show();
              {
                  send(0x04,0xc8);

              }
                  break;
              case R.id.radio43:
                   //DisplayToast("灯光亮度 40%！");
                  Toast.makeText(Firstpage.this, "第四路灯光亮度为 40%！",
                                Toast.LENGTH_SHORT).show();
              {
                  send(0x04,0x91);

              }
                  break;
              case R.id.radio44:
                    //DisplayToast("灯光亮度 60%！");
                  Toast.makeText(Firstpage.this, "第四路灯光亮度为 60%！",
                                Toast.LENGTH_SHORT).show();
              {
```

```
                send(0x04,0x5a);

            }
                break;
            case R.id.radio45:
                 // DisplayToast("灯光亮度 80%！");
                Toast.makeText(Firstpage.this, "第四路灯光亮度为 80%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x04,0x23);

            }
                break;
            default:
                 // DisplayToast("灯光亮度 100%！");
                Toast.makeText(Firstpage.this, "第四路灯光亮度为 100%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x04,0x00);

            }
                break;
            }

        }
    });



        ibutton1_o=(ImageButton) findViewById(R.id.imageButton2);
        ibutton1_o.setOnClickListener(new ClickEventKey());
        ibutton1_c=(ImageButton) findViewById(R.id.imageButton3);
        ibutton1_c.setOnClickListener(new ClickEventKey());
        ibutton2_o=(ImageButton) findViewById(R.id.imageButton4);
        ibutton2_o.setOnClickListener(new ClickEventKey());
        ibutton2_c=(ImageButton) findViewById(R.id.imageButton5);
        ibutton2_c.setOnClickListener(new ClickEventKey());
        ibutton3_o=(ImageButton) findViewById(R.id.imageButton6);
        ibutton3_o.setOnClickListener(new ClickEventKey());
        ibutton3_c=(ImageButton) findViewById(R.id.imageButton7);
        ibutton3_c.setOnClickListener(new ClickEventKey());
        ibutton4_o=(ImageButton) findViewById(R.id.imageButton8);
        ibutton4_o.setOnClickListener(new ClickEventKey());
        ibutton4_c=(ImageButton) findViewById(R.id.imageButton9);
        ibutton4_c.setOnClickListener(new ClickEventKey());
}

class ClickEventKey implements View.OnClickListener {
 @Override
 public void onClick(View v) {
 //     if (v == ibutton1_o)
```

```
switch (v.getId())

{
case R.id.imageButton2:
{
      send(0x01,0x23);
}

break;
case R.id.imageButton3:

{
      send(0x01,0xff);
}

break;
case R.id.imageButton4:
{
      send(0x02,0x23);
}

break;
case R.id.imageButton5:
{
      send(0x02,0xff);
}

break;
      case R.id.imageButton6:
{
     send(0x03,0x23);
}

break;
      case R.id.imageButton7:
{
      send(0x03,0xff);
}

break;
      case R.id.imageButton8:
{
      send(0x04,0x23);
}

break;
      case R.id.imageButton9:
{
      send(0x04,0xff);
}
```

```
            break;
        }
            }
        }
private void    send(int Room,int Grade){

            try {

                    //String strpass="@@UP...";
                    byte[ ] byteone=new byte[8];//=strpass.getBytes("US-ASCII");
                        byteone[0]=(byte)0xf5;
                        byteone[1]=(byte)0x5f;
                        byteone[2]=(byte)0x00;
                        byteone[3]=(byte)Room;
                        byteone[4]=(byte)0x03;
                        byteone[5]=(byte)0x00;
                        byteone[6]=(byte)Grade;
                        byteone[7]=(byte)0x06;

                    //byte[ ] bytekai=strpass.getBytes("US-ASCII");
                tmpOut = BluetoothMain.btSocket.getOutputStream();
                tmpOut.write(byteone);}
             catch (IOException e) {
                Log.e("BluetoothReadService", "temp sockets not created", e);
            }
            }
}
```

14.2.5　控制第二路光线的亮度

编写文件 Secondpage.java，功能是监听用户在第二路调光单选按钮列表中的选择值，根据这个值来控制光线的亮度。文件 Secondpage.java 的具体实现代码如下所示。

```
public class Secondpage extends Activity {
        private ImageButton imagebutton;
        private ImageButton ibutton5_o,ibutton5_c,ibutton6_o,ibutton6_c,ibutton7_o,
        ibutton7_c,ibutton8_o,ibutton8_c;
        RadioGroup radiogroup0,radiogroup1,radiogroup2,radiogroup3;
        RadioButton radio1,radio2,radio3,radio4,radio5,radio6;
        RadioButton radio11,radio12,radio13,radio14,radio15,radio16;
        RadioButton radio21,radio22,radio23,radio24,radio25,radio26;
        RadioButton radio31,radio32,radio33,radio34,radio35,radio36;
        OutputStream tmpOut = null;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.second);
            imagebutton=(ImageButton) findViewById(R.id.imageButton1);
            imagebutton.setOnClickListener(new Button.OnClickListener(){
                    @Override
                    public void onClick(View v)
```

```
            {
             LayoutInflater inflater = getLayoutInflater();
             View layout = inflater.inflate(R.layout.dialog,
             (ViewGroup) findViewById(R.id.dialog));

              new AlertDialog.Builder(Secondpage.this)
              .setTitle("Introduce")
              .setView(layout)
              .setPositivebutton("Return", null)
              //.setNegativeButton("取消", null)
              .show();
              } });

    radiogroup0=(RadioGroup)findViewById(R.id.radioGroup2);
   radio1=(RadioButton)findViewById(R.id.radio11);
   radio2=(RadioButton)findViewById(R.id.radio12);
   radio3=(RadioButton)findViewById(R.id.radio13);
   radio4=(RadioButton)findViewById(R.id.radio14);
   radio5=(RadioButton)findViewById(R.id.radio15);
   radio6=(RadioButton)findViewById(R.id.radio16);
   radiogroup0.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

       @Override
       public void onCheckedChanged(RadioGroup group, int checkedId) {
            // TODO Auto-generated method stub

         switch (checkedId)
         {
         case R.id.radio11:
                  /*DisplayToast("灯光亮度 0%！");*/
                Toast.makeText(Secondpage.this, "第五路灯光亮度为 0%！",
                             Toast.LENGTH_SHORT).show();
         {
                send(0x05,0xff);
         }
               break;
         case R.id.radio12:
                 // DisplayToast("灯光亮度 20%！");
                Toast.makeText(Secondpage.this, "第五路灯光亮度为 20%！",
                                   Toast.LENGTH_SHORT).show();
         {
                send(0x05,0xc8);
         }
               break;
         case R.id.radio13:
                 //DisplayToast("灯光亮度 40%！");
                Toast.makeText(Secondpage.this, "第五路灯光亮度为 40%！",
                                   Toast.LENGTH_SHORT).show();
         {
                send(0x05,0x91);
         }
               break;
```

```
            case R.id.radio14:
                   //DisplayToast("灯光亮度 60%！");
                 Toast.makeText(Secondpage.this, "第五路灯光亮度为 60%！",
                                Toast.LENGTH_SHORT).show();
            {
                 send(0x05,0x5a);
            }
                break;
            case R.id.radio15:
                  // DisplayToast("灯光亮度 80%！");
                 Toast.makeText(Secondpage.this, "第五路灯光亮度为 80%！",
                                Toast.LENGTH_SHORT).show();
            {
                 send(0x05,0x23);
            }
                break;
            default:
                  // DisplayToast("灯光亮度 100%！");
                 Toast.makeText(Secondpage.this, "第一路灯光亮度为 100%！",
                                Toast.LENGTH_SHORT).show();
            {
                 send(0x05,0x00);
            }
                break;
            }

        }
    });

    radiogroup1=(RadioGroup)findViewById(R.id.radioGroup3);
    radio11=(RadioButton)findViewById(R.id.radio21);
    radio12=(RadioButton)findViewById(R.id.radio22);
    radio13=(RadioButton)findViewById(R.id.radio23);
    radio14=(RadioButton)findViewById(R.id.radio24);
    radio15=(RadioButton)findViewById(R.id.radio25);
    radio16=(RadioButton)findViewById(R.id.radio26);
    radiogroup1.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            // TODO Auto-generated method stub

            switch (checkedId)
            {
            case R.id.radio21:
                   /*DisplayToast("灯光亮度 0%！");*/
                 Toast.makeText(Secondpage.this, "第六路灯光亮度为 0%！",
                            Toast.LENGTH_SHORT).show();
            {
                 send(0x06,0xff);
                 }
                break;
```

```
            case R.id.radio22:
                // DisplayToast("灯光亮度 20%！");
               Toast.makeText(Secondpage.this, "第六路灯光亮度为 20%！",
                                    Toast.LENGTH_SHORT).show();
            {
               send(0x06,0xc8);
               }
              break;
            case R.id.radio23:
                //DisplayToast("灯光亮度 40%！");
               Toast.makeText(Secondpage.this, "第六路灯光亮度为 40%！",
                                    Toast.LENGTH_SHORT).show();
            {
               send(0x06,0x91);
               }
              break;
            case R.id.radio24:
                 //DisplayToast("灯光亮度 60%！");
               Toast.makeText(Secondpage.this, "第六路灯光亮度为 60%！",
                                    Toast.LENGTH_SHORT).show();
            {
               send(0x06,0x5a);
               }
              break;
            case R.id.radio25:
                // DisplayToast("灯光亮度 80%！");
               Toast.makeText(Secondpage.this, "第六路灯光亮度为 80%！",
                                    Toast.LENGTH_SHORT).show();
            {
               send(0x06,0x23);
            }
              break;
            default:
                // DisplayToast("灯光亮度 100%！");
               Toast.makeText(Secondpage.this, "第六路灯光亮度为 100%！",
                                    Toast.LENGTH_SHORT).show();
            {
               send(0x06,0x00);
            }
              break;
            }

        }
    });


    radiogroup2=(RadioGroup)findViewById(R.id.radioGroup4);
    radio21=(RadioButton)findViewById(R.id.radio31);
    radio22=(RadioButton)findViewById(R.id.radio32);
    radio23=(RadioButton)findViewById(R.id.radio33);
    radio24=(RadioButton)findViewById(R.id.radio34);
    radio25=(RadioButton)findViewById(R.id.radio35);
```

```
radio26=(RadioButton)findViewById(R.id.radio36);
radiogroup2.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        // TODO Auto-generated method stub

      switch (checkedId)
      {
      case R.id.radio31:
            /*DisplayToast("灯光亮度 0%！");*/
          Toast.makeText(Secondpage.this, "第七路灯光亮度为 0%！",
                    Toast.LENGTH_SHORT).show();
      {

          send(0x07,0xff);
          }
         break;
      case R.id.radio32:
           // DisplayToast("灯光亮度 20%！");
          Toast.makeText(Secondpage.this, "第七路灯光亮度为 20%！",
                         Toast.LENGTH_SHORT).show();
      {
          send(0x07,0xc8);
          }
         break;
      case R.id.radio33:
           //DisplayToast("灯光亮度 40%！");
          Toast.makeText(Secondpage.this, "第七路灯光亮度为 40%！",
                         Toast.LENGTH_SHORT).show();
      {
          send(0x07,0x91);
          }
         break;
      case R.id.radio34:
            //DisplayToast("灯光亮度 60%！");
          Toast.makeText(Secondpage.this, "第七路灯光亮度为 60%！",
                         Toast.LENGTH_SHORT).show();
      {
          send(0x07,0x5a);
          }
         break;
      case R.id.radio35:
           // DisplayToast("灯光亮度 80%！");
          Toast.makeText(Secondpage.this, "第七路灯光亮度为 80%！",
                         Toast.LENGTH_SHORT).show();
      {
          send(0x07,0x23);
          }
         break;
      default:
           // DisplayToast("灯光亮度 100%！");
```

```
                Toast.makeText(Secondpage.this, "第七路灯光亮度为 100%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x07,0x00);
            }
                break;
            }

        }
    });    //第四路


    radiogroup3=(RadioGroup)findViewById(R.id.radioGroup5);
    radio31=(RadioButton)findViewById(R.id.radio41);
    radio32=(RadioButton)findViewById(R.id.radio42);
    radio33=(RadioButton)findViewById(R.id.radio43);
    radio34=(RadioButton)findViewById(R.id.radio44);
    radio35=(RadioButton)findViewById(R.id.radio45);
    radio36=(RadioButton)findViewById(R.id.radio46);
    radiogroup3.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            // TODO Auto-generated method stub

            switch (checkedId)
            {
            case R.id.radio41:
                    /*DisplayToast("灯光亮度 0%！");*/
                Toast.makeText(Secondpage.this, "第八路灯光亮度为 0%！",
                            Toast.LENGTH_SHORT).show();
            {
                send(0x08,0xff);
            }
                break;
            case R.id.radio42:
                    // DisplayToast("灯光亮度 20%！");
                Toast.makeText(Secondpage.this, "第八路灯光亮度为 20%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x08,0xc8);
                }
                break;
            case R.id.radio43:
                    //DisplayToast("灯光亮度 40%！");
                Toast.makeText(Secondpage.this, "第八路灯光亮度为 40%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x08,0x91);
                }
                break;
            case R.id.radio44:
```

```
                    //DisplayToast("灯光亮度 60%！");
                  Toast.makeText(Secondpage.this, "第八路灯光亮度为 60%！",
                                 Toast.LENGTH_SHORT).show();
            {
                  send(0x08,0x5a);
                  }
                 break;
            case R.id.radio45:
                   // DisplayToast("灯光亮度 80%！");
                  Toast.makeText(Secondpage.this, "第八路灯光亮度为 80%！",
                                 Toast.LENGTH_SHORT).show();
            {
                  send(0x08,0x23);
                  }
                 break;
            default:
                   // DisplayToast("灯光亮度 100%！");
                  Toast.makeText(Secondpage.this, "第八路灯光亮度为 100%！",
                                 Toast.LENGTH_SHORT).show();
            {
                  send(0x08,0x00);
                  }
                 break;
            }

          }
      });

       ibutton5_o=(ImageButton) findViewById(R.id.imageButton2);
       ibutton5_o.setOnClickListener(new ClickEventKey());
       ibutton5_c=(ImageButton) findViewById(R.id.imageButton3);
       ibutton5_c.setOnClickListener(new ClickEventKey());
       ibutton6_o=(ImageButton) findViewById(R.id.imageButton4);
       ibutton6_o.setOnClickListener(new ClickEventKey());
       ibutton6_c=(ImageButton) findViewById(R.id.imageButton5);
       ibutton6_c.setOnClickListener(new ClickEventKey());
       ibutton7_o=(ImageButton) findViewById(R.id.imageButton6);
       ibutton7_o.setOnClickListener(new ClickEventKey());
       ibutton7_c=(ImageButton) findViewById(R.id.imageButton7);
       ibutton7_c.setOnClickListener(new ClickEventKey());
       ibutton8_o=(ImageButton) findViewById(R.id.imageButton8);
       ibutton8_o.setOnClickListener(new ClickEventKey());
       ibutton8_c=(ImageButton) findViewById(R.id.imageButton9);
       ibutton8_c.setOnClickListener(new ClickEventKey());
 }

class ClickEventKey implements View.OnClickListener {
 @Override
 public void onClick(View v) {
 //    if (v == ibutton1_o)
       switch (v.getId())
```

```
        {
        case R.id.imageButton2:
        {
            send(0x05,0x23);
            }

        break;
        case R.id.imageButton3:
            {
            send(0x05,0xff);
        }

        break;
        case R.id.imageButton4:
        {
            send(0x06,0x23);
            }

        break;
        case R.id.imageButton5:
        {
            send(0x06,0xff);
        }

        break;
            case R.id.imageButton6:
        {
            send(0x07,0x23);
        }

        break;
            case R.id.imageButton7:
        {
            send(0x07,0xff);
        }

        break;
            case R.id.imageButton8:
        {
            send(0x08,0x23);
        }

        break;
            case R.id.imageButton9:
        {
            send(0x08,0xff);
        }

        break;
        }
        }
    }
```

```
private void    send(int Room,int Grade){

             try {

                     //String strpass="@@UP...";
                    byte[ ] byteone=new byte[8];//=strpass.getBytes("US-ASCII");
                         byteone[0]=(byte)0xf5;
                         byteone[1]=(byte)0x5f;
                         byteone[2]=(byte)0x00;
                         byteone[3]=(byte)Room;
                         byteone[4]=(byte)0x03;
                         byteone[5]=(byte)0x00;
                         byteone[6]=(byte)Grade;
                         byteone[7]=(byte)0x06;

                      //byte[ ] bytekai=strpass.getBytes("US-ASCII");
                  tmpOut = BluetoothMain.btSocket.getOutputStream();
                  tmpOut.write(byteone);}
              catch (IOException e) {
                  Log.e("BluetoothReadService", "temp sockets not created", e);
             }
              }
}
```

到此为止，本实例的主要功能模块的实现过程介绍完毕。本实例执行后的效果如图 14-1 所示。



图 14-1　执行效果

第 15 章　智能闹钟系统

当今社会人们的生活节奏越来越快，随着硬件移动设备越来越先进，人们对时间的观念也越来越高。Android 作为一个开源的智能手机系统，具备了闹钟的功能。和传统的闹钟相比，Android 手机闹钟更加灵活、方便、准时。在本章的内容中，将详细讲解在 Android 系统中开发闹钟系统的方法，了解大型闹钟系统的具体开发流程。

15.1 项 目 介 绍

知识点讲解：光盘:视频\知识点\第 15 章\项目介绍.avi

本章播放器源码保存在本书附带光盘中的“光盘:\daima\15”目录下。在讲解具体编码之前，先简要介绍本项目的产生背景和项目的知识，为后面的具体编码打好理论基础。

15.1.1 系统需求分析

在快节奏的生活中，闹钟已经成为广大人群不可缺少的物品。Android 作为一款强大的智能手机操作系统，内置了闹钟功能，但是为了满足人们的更高要求，开发一个个性化的、功能更强大的闹钟系统势在必行。

根据市场调研分析，一个典型的闹钟系统应该具备如下所示的功能。

（1）添加闹钟

向系统中添加新的闹钟，设置什么时候闹钟会提醒我们，时间会精确到几点几分。

（2）管理闹钟

为了提高设置闹钟的效率，可以在原有已经设置的闹钟的基础上管理闹钟。具体来说，可以修改闹钟的具体时间和其他属性（例如震动、铃声、重复次数）。

（3）删除闹钟

对于系统中不需要的闹钟，可以及时删除，节约系统空间。

（4）设置其他属性

在设置一个闹钟时，除了设置一个具体时间外，还可以设置重复次数、闹钟开关、在哪一天重复、铃声、震动和标记。

15.1.2 构成模块

根据前面的系统需求分析，可以规划出本系统的构成模块如图 15-1 所示。

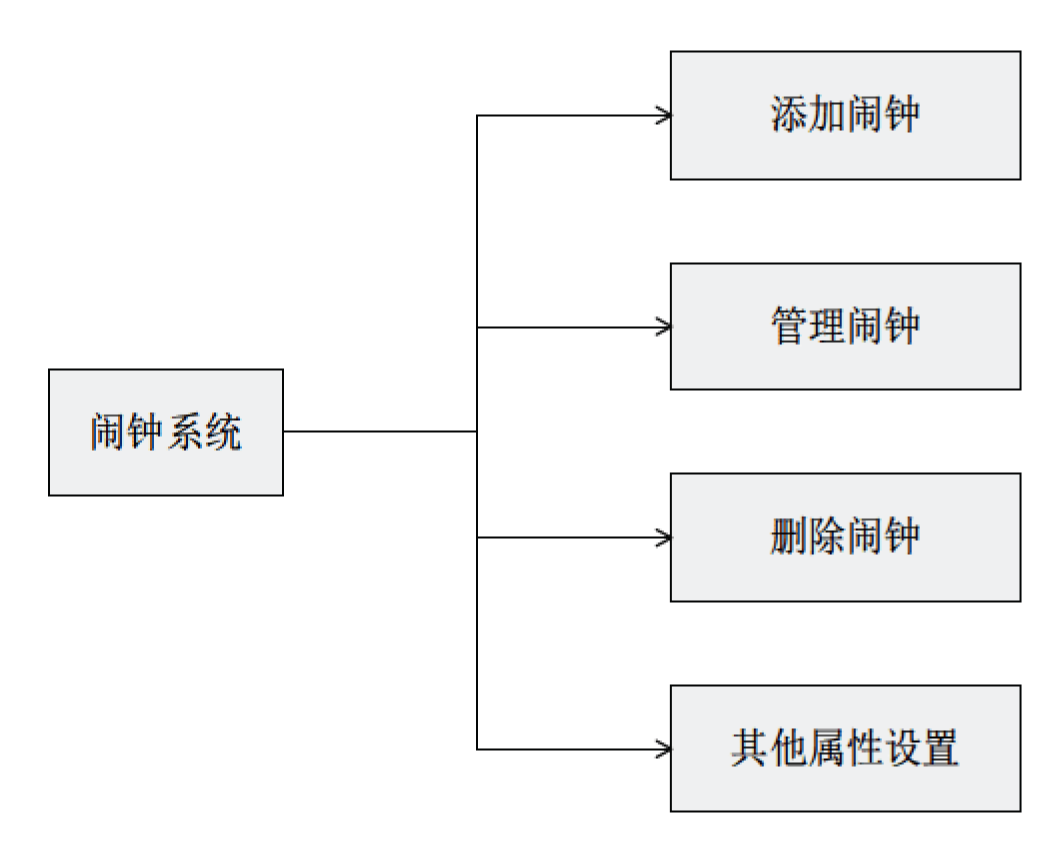


图 15-1　系统构成模块

15.2　系统主界面

知识点讲解：光盘:视频\知识点\第 15 章\系统主界面.avi

本系统的主界面比较简单，执行后会在屏幕中间显示当前的时间，并显示最近的闹钟和天气情况，并在屏幕下方显示一个操作导航。执行效果如图 15-2 所示。



图 15-2　执行效果

在本节的内容中，将详细讲解主界面的具体实现过程。

15.2.1　布局文件

主界面的核心布局功能是通过文件 desk_clock_buttons.xml 实现的，主要实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
    android:layout_weight="0"
    >
    <ImageButton android:id="@+id/alarm_button"
            style="@style/ButtonStripLeft"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_weight=".25"
        android:src="@drawable/ic_clock_strip_alarm"
        android:contentDescription="@string/alarm_button_description"
        />
    <ImageButton android:id="@+id/gallery_button"
            style="@style/ButtonStripMiddle"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_weight=".25"
        android:src="@drawable/ic_clock_strip_gallery"
        android:contentDescription="@string/gallery_button_description"
        />
    <ImageButton android:id="@+id/music_button"
            style="@style/ButtonStripMiddle"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_weight=".25"
        android:src="@drawable/ic_clock_strip_music"
        android:contentDescription="@string/music_button_description"
        />
    <ImageButton android:id="@+id/home_button"
            style="@style/ButtonStripRight"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_weight=".25"
        android:src="@drawable/ic_clock_strip_home"
        android:contentDescription="@string/home_button_description"
        />
</LinearLayout>
```

本系统执行后，展示在用户面前的便是图 15-2 所示的界面。

15.2.2 程序文件

系统主界面的程序文件是 DeskClock，具体实现流程如下所示。

（1）定义类 DeskClock，并定义需要常量，具体实现代码如下所示。

```
public class DeskClock extends Activity {
    private static final boolean DEBUG = false;

    private static final String LOG_TAG = "DeskClock";

    //午夜闹钟动作（可以更新日期显示）.
    private static final String ACTION_MIDNIGHT = "com.android.superdeskclock.MIDNIGHT";
```

```
    // 设置天气和时间的间隔
    private final long QUERY_WEATHER_DELAY = 60 * 60 * 1000; // 1 hr

    //为闹钟设置广播
    private static final String DOCK_SETTINGS_ACTION = "com.android.settings.DOCK_SETTINGS";

    // 设置闹钟延迟设置 5 分钟
    private final long SCREEN_SAVER_TIMEOUT = 5 * 60 * 1000; // 5 min

    //设置屏幕保护程序复位延迟
    private final long SCREEN_SAVER_MOVE_DELAY = 60 * 1000; // 1 min

    //设置在屏幕保护模式时的文本和图形的颜色
    private final int SCREEN_SAVER_COLOR = 0xFF308030;
    private final int SCREEN_SAVER_COLOR_DIM = 0xFF183018;

    // 设置时钟显示壁纸与黑色图层之间的不透明度
    private final float DIM_BEHIND_AMOUNT_NORMAL = 0.4f;
    // 高对比变暗
    private final float DIM_BEHIND_AMOUNT_DIMMED = 0.8f;

    //下面是内部消息 ID
    private final int QUERY_WEATHER_DATA_MSG          = 0x1000;
    private final int UPDATE_WEATHER_DISPLAY_MSG      = 0x1001;
    private final int SCREEN_SAVER_TIMEOUT_MSG        = 0x2000;
    private final int SCREEN_SAVER_MOVE_MSG           = 0x2001;

    //下面是天气预报查询信息设置
    private static final String GENIE_PACKAGE_ID = "com.google.android.apps.genie.geniewidget";
    private static final String WEATHER_CONTENT_AUTHORITY = GENIE_PACKAGE_ID + ".weather";
    private static final String WEATHER_CONTENT_PATH = "/weather/current";
    private static final String[ ] WEATHER_CONTENT_COLUMNS = new String[ ] {
            "location",
            "timestamp",
            "temperature",
            "highTemperature",
            "lowTemperature",
            "iconUrl",
            "iconResId",
            "description",
        };
```

（2）定义函数 moveScreenSaverTo()，功能是移动系统的屏幕保护程序，其中参数 x 和 y 表示位置坐标。函数 moveScreenSaverTo()的具体实现代码如下所示。

```
private void moveScreenSaverTo(int x, int y) {
    if (!mScreenSaverMode) return;

    final View saver_view = findViewById(R.id.saver_view);
```

```
        DisplayMetrics metrics = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(metrics);

        if (x < 0 || y < 0) {
            int myWidth = saver_view.getMeasuredWidth();
            int myHeight = saver_view.getMeasuredHeight();
            x = (int)(mRNG.nextFloat()*(metrics.widthPixels - myWidth));
            y = (int)(mRNG.nextFloat()*(metrics.heightPixels - myHeight));
        }

        if (DEBUG) Log.d(LOG_TAG, String.format("screen saver: %d: jumping to (%d,%d)",
                System.currentTimeMillis(), x, y));

        saver_view.setLayoutParams(new AbsoluteLayout.LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,
            ViewGroup.LayoutParams.WRAP_CONTENT,
            x,
            y));

        // Synchronize our jumping so that it happens exactly on the second
        mHandy.sendEmptyMessageDelayed(SCREEN_SAVER_MOVE_MSG,
            SCREEN_SAVER_MOVE_DELAY +
            (1000 - (System.currentTimeMillis() % 1000)));
    }
```

（3）定义函数 setWakeLock()，功能是设置一个唤醒锁，到闹钟时间时会唤醒。函数 setWakeLock()的具体实现代码如下所示。

```
    private void setWakeLock(boolean hold) {
        if (DEBUG) Log.d(LOG_TAG, (hold ? "hold" : " releas") + "ing wake lock");
        Window win = getWindow();
        WindowManager.LayoutParams winParams = win.getAttributes();
        winParams.flags |= (WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD
                | WindowManager.LayoutParams.FLAG_SHOW_WHEN_LOCKED
//                  | WindowManager.LayoutParams.FLAG_ALLOW_LOCK_WHILE_SCREEN_ON
                | WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON);
        if (hold)
            winParams.flags |= WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON;
        else
            winParams.flags &= (~WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        win.setAttributes(winParams);
    }
```

（4）定义函数 scheduleScreenSaver()，功能是设置屏幕保护程序的时间表，即何时进入屏保状态。定义函数 restoreScreen()，功能是恢复到正常的屏幕模式。定义函数 saveScreen()，用于保存当前的屏幕状态，此函数能够处理 OLED 模式的屏幕保护程序。上述 3 个函数的具体实现代码如下所示。

```
    private void scheduleScreenSaver() {
        // reschedule screen saver
        mHandy.removeMessages(SCREEN_SAVER_TIMEOUT_MSG);
        mHandy.sendMessageDelayed(
            Message.obtain(mHandy, SCREEN_SAVER_TIMEOUT_MSG),
            SCREEN_SAVER_TIMEOUT);
```



```
}

private void restoreScreen() {
    if (!mScreenSaverMode) return;
    if (DEBUG) Log.d(LOG_TAG, "restoreScreen");
    mScreenSaverMode = false;
    initViews();
    doDim(false); // restores previous dim mode
    // policy: update weather info when returning from screen saver
    if (mPluggedIn) requestWeatherDataFetch();

    scheduleScreenSaver();

    refreshAll();
}
private void saveScreen() {
    if (mScreenSaverMode) return;
    if (DEBUG) Log.d(LOG_TAG, "saveScreen");

    //快藏起来的 X/Y 的当前日期
    final View oldTimeDate = findViewById(R.id.time_date);
    int oldLoc[ ] = new int[2];
    oldTimeDate.getLocationOnScreen(oldLoc);

    mScreenSaverMode = true;
    Window win = getWindow();
    WindowManager.LayoutParams winParams = win.getAttributes();
    winParams.flags |= WindowManager.LayoutParams.FLAG_FULLSCREEN;
    win.setAttributes(winParams);

    //在切换布局屏幕时放弃任何内部聚焦
    final View focused = getCurrentFocus();
    if (focused != null) focused.clearFocus();

    setContentView(R.layout.desk_clock_saver);

    mTime = (DigitalClock) findViewById(R.id.time);
    mDate = (TextView) findViewById(R.id.date);
    mNextAlarm = (TextView) findViewById(R.id.nextAlarm);

    final int color = mDimmed ? SCREEN_SAVER_COLOR_DIM : SCREEN_SAVER_COLOR;

    ((TextView)findViewById(R.id.timeDisplay)).setTextColor(color);
    ((TextView)findViewById(R.id.am_pm)).setTextColor(color);
    mDate.setTextColor(color);
    mNextAlarm.setTextColor(color);
    mNextAlarm.setCompoundDrawablesWithIntrinsicBounds(
        getResources().getDrawable(mDimmed
            ? R.drawable.ic_lock_idle_alarm_saver_dim
            : R.drawable.ic_lock_idle_alarm_saver),
        null, null, null);
```

```
        mBatteryDisplay =
        mWeatherCurrentTemperature =
        mWeatherHighTemperature =
        mWeatherLowTemperature =
        mWeatherLocation = null;
        mWeatherIcon = null;

        refreshDate();
        refreshAlarm();

        moveScreenSaverTo(oldLoc[0], oldLoc[1]);
}
```

（5）在系统的主界面中显示了当日的天气情况，此功能通过如下函数实现。

- 函数requestWeatherDataFetch()：功能是获取天气数据。
- 函数scheduleWeatherQueryDelayed()：功能是处理天气查询延迟时间。
- 函数queryWeatherData()：功能是查询天气数据。
- 函数updateWeatherDisplay()：功能是在界面中更新天气显示。

上述函数的具体实现代码如下所示。

```
//告诉 widget 部件从网络载入新的数据
private void requestWeatherDataFetch() {
        if (DEBUG) Log.d(LOG_TAG, "forcing the Genie widget to update weather now...");
        sendBroadcast(new Intent(ACTION_GENIE_REFRESH).putExtra("requestWeather", true));
        // we expect the result to show up in our content observer
}

private boolean supportsWeather() {
        return (mGenieResources != null);
}

private void scheduleWeatherQueryDelayed(long delay) {
        // cancel any existing scheduled queries
        unscheduleWeatherQuery();
        if (DEBUG) Log.d(LOG_TAG, "scheduling weather fetch message for " + delay + "ms from now");
        mHandy.sendEmptyMessageDelayed(QUERY_WEATHER_DATA_MSG, delay);
}
private void unscheduleWeatherQuery() {
        mHandy.removeMessages(QUERY_WEATHER_DATA_MSG);
}

private void queryWeatherData() {
        // if we couldn't load the weather widget's resources, we simply
        // assume it's not present on the device.
        if (mGenieResources == null) return;

        Uri queryUri = new Uri.Builder()
                .scheme(android.content.ContentResolver.SCHEME_CONTENT)
                .authority(WEATHER_CONTENT_AUTHORITY)
                .path(WEATHER_CONTENT_PATH)
```

```
        .appendPath(new Long(System.currentTimeMillis()).toString())
        .build();

if (DEBUG) Log.d(LOG_TAG, "querying genie: " + queryUri);

Cursor cur;
try {
    cur = getContentResolver().query(
        queryUri,
        WEATHER_CONTENT_COLUMNS,
        null,
        null,
        null);
} catch (RuntimeException e) {
    Log.e(LOG_TAG, "Weather query failed", e);
    cur = null;
}

if (cur != null && cur.moveToFirst()) {
    if (DEBUG) {
        java.lang.StringBuilder sb =
            new java.lang.StringBuilder("Weather query result: {");
        for(int i=0; i<cur.getColumnCount(); i++) {
            if (i>0) sb.append(", ");
            sb.append(cur.getColumnName(i))
                .append("=")
                .append(cur.getString(i));
        }
        sb.append("}");
        Log.d(LOG_TAG, sb.toString());
    }

    mWeatherIconDrawable = mGenieResources.getDrawable(cur.getInt(
        cur.getColumnIndexOrThrow("iconResId")));

    mWeatherLocationString = cur.getString(
        cur.getColumnIndexOrThrow("location"));

    // any of these may be NULL
    final int colTemp = cur.getColumnIndexOrThrow("temperature");
    final int colHigh = cur.getColumnIndexOrThrow("highTemperature");
    final int colLow = cur.getColumnIndexOrThrow("lowTemperature");

    mWeatherCurrentTemperatureString =
        cur.isNull(colTemp)
            ? "\u2014"
            : String.format("%d\u00b0", cur.getInt(colTemp));
    mWeatherHighTemperatureString =
        cur.isNull(colHigh)
            ? "\u2014"
            : String.format("%d\u00b0", cur.getInt(colHigh));
```

```
            mWeatherLowTemperatureString =
                cur.isNull(colLow)
                    ? "\u2014"
                    : String.format("%d\u00b0", cur.getInt(colLow));
        } else {
            Log.w(LOG_TAG, "No weather information available (cur="
                + cur +")");
            mWeatherIconDrawable = null;
            mWeatherLocationString = getString(R.string.weather_fetch_failure);
            mWeatherCurrentTemperatureString =
                mWeatherHighTemperatureString =
                mWeatherLowTemperatureString = "";
        }

        if (cur != null) {
            // clean up cursor
            cur.close();
        }

        mHandy.sendEmptyMessage(UPDATE_WEATHER_DISPLAY_MSG);
}

private void refreshWeather() {
        if (supportsWeather())
            scheduleWeatherQueryDelayed(0);
        updateWeatherDisplay(); // in case we have it cached
}

private void updateWeatherDisplay() {
        if (mWeatherCurrentTemperature == null) return;

        mWeatherCurrentTemperature.setText(mWeatherCurrentTemperatureString);
        mWeatherHighTemperature.setText(mWeatherHighTemperatureString);
        mWeatherLowTemperature.setText(mWeatherLowTemperatureString);
        mWeatherLocation.setText(mWeatherLocationString);
        mWeatherIcon.setImageDrawable(mWeatherIconDrawable);
}
```

（6）在系统的主界面中显示了电量信息，此功能通过如下函数实现。

- 函数handleBatteryUpdate()：功能是更新显示电量信息。
- 函数refreshBattery()：功能是刷新系统的当前电量。

上述函数的具体实现代码如下所示。

```
private void handleBatteryUpdate(int plugStatus, int batteryLevel) {
        final boolean pluggedIn = (plugStatus == BATTERY_STATUS_CHARGING || plugStatus == BATTERY_
STATUS_FULL);
        if (pluggedIn != mPluggedIn) {
            setWakeLock(pluggedIn);

            if (pluggedIn) {
                // policy: update weather info when attaching to power
                requestWeatherDataFetch();
```

```
            }
        }
        if (pluggedIn != mPluggedIn || batteryLevel != mBatteryLevel) {
            mBatteryLevel = batteryLevel;
            mPluggedIn = pluggedIn;
            refreshBattery();
        }
}

private void refreshBattery() {
        if (mBatteryDisplay == null) return;

        if (mPluggedIn /* || mBatteryLevel < LOW_BATTERY_THRESHOLD */) {
            mBatteryDisplay.setCompoundDrawablesWithIntrinsicBounds(
                0, 0, android.R.drawable.ic_lock_idle_charging, 0);
            mBatteryDisplay.setText(
                getString(R.string.battery_charging_level, mBatteryLevel));
            mBatteryDisplay.setVisibility(View.VISIBLE);
        } else {
            mBatteryDisplay.setVisibility(View.INVISIBLE);
        }
}
```

（7）为了及时更新系统主界面，特意提供了如下刷新函数。

- 函数refreshDate()：功能是更新显示当前的时间。
- 函数refreshAlarm()：功能是刷新显示系统的闹钟。
- 函数refreshAll()：功能是分别调用时间、闹钟、电量和天气这4个刷新函数。

上述函数的具体实现代码如下所示。

```
private void refreshDate() {
        final Date now = new Date();
        if (DEBUG) Log.d(LOG_TAG, "refreshing date..." + now);
        mDate.setText(DateFormat.format(mDateFormat, now));
}

private void refreshAlarm() {
        if (mNextAlarm == null) return;

        String nextAlarm = Settings.System.getString(getContentResolver(),
                Settings.System.NEXT_ALARM_FORMATTED);
        if (!TextUtils.isEmpty(nextAlarm)) {
            mNextAlarm.setText(nextAlarm);
            //mNextAlarm.setCompoundDrawablesWithIntrinsicBounds(
            //      android.R.drawable.ic_lock_idle_alarm, 0, 0, 0);
            mNextAlarm.setVisibility(View.VISIBLE);
        } else {
            mNextAlarm.setVisibility(View.INVISIBLE);
        }
}

private void refreshAll() {
```

```
        refreshDate();
        refreshAlarm();
        refreshBattery();
        refreshWeather();
}
```

（8）定义函数 doDim()，设置系统进入模糊模式显示状态。在系统从休眠模式恢复到正常模式时，之间的模式就是模糊模式。函数 doDim()的具体实现代码如下所示。

```
private void doDim(boolean fade) {
        View tintView = findViewById(R.id.window_tint);
        if (tintView == null) return;

        Window win = getWindow();
        WindowManager.LayoutParams winParams = win.getAttributes();

        winParams.flags |= (WindowManager.LayoutParams.FLAG_LAYOUT_IN_SCREEN);
        winParams.flags |= (WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);

        // dim the wallpaper somewhat (how much is determined below)
        winParams.flags |= (WindowManager.LayoutParams.FLAG_DIM_BEHIND);

        if (mDimmed) {
                winParams.flags |= WindowManager.LayoutParams.FLAG_FULLSCREEN;
                winParams.dimAmount = DIM_BEHIND_AMOUNT_DIMMED;
//                      winParams.buttonBrightness = WindowManager.LayoutParams.BRIGHTNESS_OVERRIDE_OFF;

                // show the window tint
                tintView.startAnimation(AnimationUtils.loadAnimation(this,
                        fade ? R.anim.dim
                             : R.anim.dim_instant));
        } else {
                winParams.flags &= (~WindowManager.LayoutParams.FLAG_FULLSCREEN);
                winParams.dimAmount = DIM_BEHIND_AMOUNT_NORMAL;
//                      winParams.buttonBrightness = WindowManager.LayoutParams.BRIGHTNESS_OVERRIDE_NONE;

                // hide the window tint
                tintView.startAnimation(AnimationUtils.loadAnimation(this,
                        fade ? R.anim.undim
                             : R.anim.undim_instant));
        }

        win.setAttributes(winParams);
}
```

（9）为了保证在系统主界面中实现不同模式的转换，特意提供了如下模式转换函数。

- 函数initViews()：功能是初始化视图界面。
- 函数onPause()：功能是暂停当前的模式转换，关闭屏幕保护程序，并取消任何超时操作，但取消模糊模式除外。
- 函数onStop()：功能是停止当前的模式转换。
- 函数onStart()：功能是开启系统界面，分别调用对应的函数显示时间、闹钟、电量和天气。



- 函数onNewIntent()：功能是开启新的界面视图。
- 函数onConfigurationChanged()：功能是根据配置改变显示视图。

上述函数的具体实现代码如下所示。

```
public void onNewIntent(Intent newIntent) {
    super.onNewIntent(newIntent);
    if (DEBUG) Log.d(LOG_TAG, "onNewIntent with intent: " + newIntent);

    // update our intent so that we can consult it to determine whether or
    // not the most recent launch was via a dock event
    setIntent(newIntent);
}

@Override
public void onStart() {
    super.onStart();

    IntentFilter filter = new IntentFilter();
    filter.addAction(Intent.ACTION_DATE_CHANGED);
    filter.addAction(Intent.ACTION_BATTERY_CHANGED);
//          filter.addAction(UiModeManager.ACTION_EXIT_DESK_MODE);
    filter.addAction(ACTION_MIDNIGHT);
    registerReceiver(mIntentReceiver, filter);
}

@Override
public void onStop() {
    super.onStop();

    unregisterReceiver(mIntentReceiver);
}

@Override
public void onResume() {
    super.onResume();
    if (DEBUG) Log.d(LOG_TAG, "onResume with intent: " + getIntent());

    // reload the date format in case the user has changed settings
    // recently
    mDateFormat = getString(R.string.full_wday_month_day_no_year);

    // Listen for updates to weather data
    Uri weatherNotificationUri = new Uri.Builder()
        .scheme(android.content.ContentResolver.SCHEME_CONTENT)
        .authority(WEATHER_CONTENT_AUTHORITY)
        .path(WEATHER_CONTENT_PATH)
        .build();
    getContentResolver().registerContentObserver(
        weatherNotificationUri, true, mContentObserver);

    // Elaborate mechanism to find out when the day rolls over
```

```
        Calendar today = Calendar.getInstance();
        today.set(Calendar.HOUR_OF_DAY, 0);
        today.set(Calendar.MINUTE, 0);
        today.set(Calendar.SECOND, 0);
        today.add(Calendar.DATE, 1);
        long alarmTimeUTC = today.getTimeInMillis();

        mMidnightIntent = PendingIntent.getBroadcast(this, 0, new Intent(ACTION_MIDNIGHT), 0);
        AlarmManager am = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
        am.setRepeating(AlarmManager.RTC, alarmTimeUTC, AlarmManager.INTERVAL_DAY, mMidnightIntent);
        if (DEBUG) Log.d(LOG_TAG, "set repeating midnight event at UTC: "
            + alarmTimeUTC + " ("
            + (alarmTimeUTC - System.currentTimeMillis())
            + " ms from now) repeating every "
            + AlarmManager.INTERVAL_DAY + " with intent: " + mMidnightIntent);

        // If we weren't previously visible but now we are, it's because we're
        // being started from another activity. So it's OK to un-dim.
        if (mTime != null && mTime.getWindowVisibility() != View.VISIBLE) {
            mDimmed = false;
        }

        // Adjust the display to reflect the currently chosen dim mode.
        doDim(false);

        restoreScreen(); // disable screen saver
        refreshAll(); // will schedule periodic weather fetch

        setWakeLock(mPluggedIn);

        scheduleScreenSaver();

        final boolean launchedFromDock
            = getIntent().hasCategory(Intent.CATEGORY_DESK_DOCK);

        if (supportsWeather() && launchedFromDock && !mLaunchedFromDock) {
            // policy: fetch weather if launched via dock connection
            if (DEBUG) Log.d(LOG_TAG, "Device now docked; forcing weather to refresh right now");
            requestWeatherDataFetch();
        }

        mLaunchedFromDock = launchedFromDock;
    }

    @Override
    public void onPause() {
        if (DEBUG) Log.d(LOG_TAG, "onPause");

        // Turn off the screen saver and cancel any pending timeouts.
        // (But don't un-dim.)
        mHandy.removeMessages(SCREEN_SAVER_TIMEOUT_MSG);
```

```
        restoreScreen();

        // Other things we don't want to be doing in the background.
        // NB: we need to keep our broadcast receiver alive in case the dock
        // is disconnected while the screen is off
        getContentResolver().unregisterContentObserver(mContentObserver);

        AlarmManager am = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
        am.cancel(mMidnightIntent);
        unscheduleWeatherQuery();

        super.onPause();
}

private void initViews() {
        //放弃任何在当前布局中的切换操作
        final View focused = getCurrentFocus();
        if (focused != null) focused.clearFocus();

        setContentView(R.layout.desk_clock);

        mTime = (DigitalClock) findViewById(R.id.time);
        mDate = (TextView) findViewById(R.id.date);
        mBatteryDisplay = (TextView) findViewById(R.id.battery);

        mTime.getRootView().requestFocus();

        mWeatherCurrentTemperature = (TextView) findViewById(R.id.weather_temperature);
        mWeatherHighTemperature = (TextView) findViewById(R.id.weather_high_temperature);
        mWeatherLowTemperature = (TextView) findViewById(R.id.weather_low_temperature);
        mWeatherLocation = (TextView) findViewById(R.id.weather_location);
        mWeatherIcon = (ImageView) findViewById(R.id.weather_icon);

        final View.OnClickListener alarmClickListener = new View.OnClickListener() {
                public void onClick(View v) {
                        startActivity(new Intent(DeskClock.this, AlarmClock.class));
                }
        };
public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        if (mScreenSaverMode) {
                moveScreenSaver();
        } else {
                initViews();
                doDim(false);
                refreshAll();
        }
}
```

（10）在主屏幕底部生成 4 个选项，选择不同的选项后会实现对应的操作，此功能的实现函数如下。

- 函数onOptionsItemSelected()：功能是根据用户选择的选项启动对应的视图界面。
- 函数onCreateOptionsMenu()：功能是创建屏幕底部的选项菜单。

❑ 函数onCreate()：功能是创建菜单视图。

上述函数的具体实现代码如下所示。

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item_alarms:
            startActivity(new Intent(DeskClock.this, AlarmClock.class));
            return true;
        case R.id.menu_item_add_alarm:
            startActivity(new Intent(this, SetAlarm.class));
            return true;
        case R.id.menu_item_dock_settings:
            startActivity(new Intent(DOCK_SETTINGS_ACTION));
            return true;
        default:
            return false;
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.desk_clock_menu, menu);
    return true;
}

@Override
protected void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    mRNG = new Random();

    try {
        mGenieResources = getPackageManager().getResourcesForApplication(GENIE_PACKAGE_ID);
    } catch (PackageManager.NameNotFoundException e) {
        // no weather info available
        Log.w(LOG_TAG, "Can't find "+GENIE_PACKAGE_ID+". Weather forecast will not be available.");
    }

    initViews();
}
```

15.3 闹钟列表模块

知识点讲解：光盘:视频\知识点\第 15 章\闹钟列表模块.avi

当单击系统主界面底部导航菜单中的图标后，会进入闹钟列表界面。执行效果如图 15-3 所示。

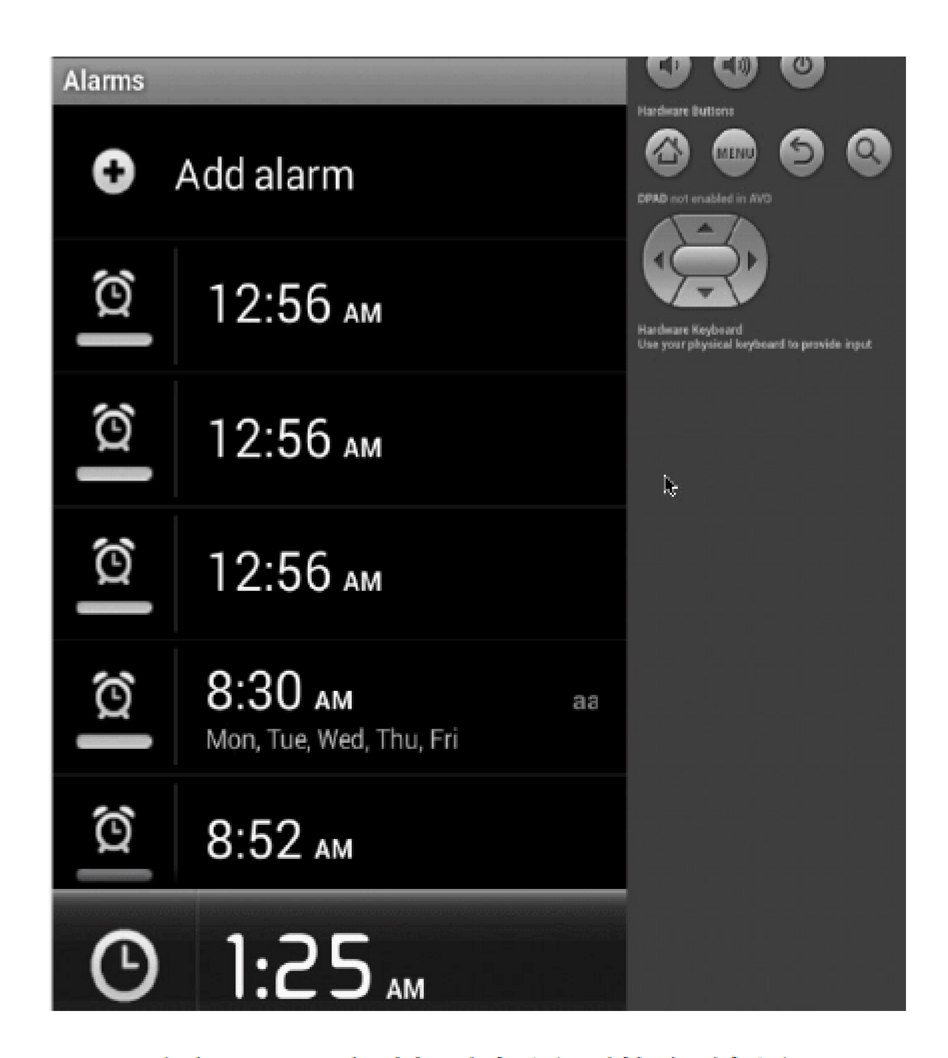


图 15-3　闹钟列表界面执行效果

在本节的内容中，将详细讲解本系统闹钟列表模块的具体实现过程。

15.3.1　设置主界面

图 15-3 所示的是闹钟列表界面，在顶部显示一个添加按钮图标。在中间显示了一个两列列表，左侧列表显示了系统中的所有闹钟的“是否可用”开关，在右侧列表显示了各个闹钟的具体时间。在底部显示了系统的当前时间。设置主界面的布局文件是 alarm_clock.xml，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/base_layout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <LinearLayout android:id="@+id/add_alarm"
        android:clickable="true"
        android:focusable="true"
        android:background="@android:drawable/list_selector_background"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <ImageView
            style="@style/alarm_list_left_column"
            android:duplicateParentState="true"
            android:gravity="center"
            android:scaleType="center"
            android:src="@drawable/add_alarm" />

        <TextView
            android:duplicateParentState="true"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
```

```
            android:layout_gravity="center_vertical"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="?android:attr/textColorPrimary"
            android:text="@string/add_alarm" />

    </LinearLayout>

    <ImageView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:scaleType="fitXY"
        android:gravity="fill_horizontal"
        android:src="@android:drawable/divider_horizontal_dark" />

    <ListView
        android:id="@+id/alarms_list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <ImageButton android:id="@+id/desk_clock_button"
            style="@style/ButtonStripLeft"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_clock_strip_desk_clock"
            android:contentDescription="@string/desk_clock_button_description"/>

        <com.android.superdeskclock.DigitalClock
            style="@style/ButtonStripRight"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent">

            <LinearLayout
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:baselineAligned="true">

                <TextView android:id="@+id/timeDisplay"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:paddingRight="6dip"
                    android:textSize="48sp"
                    android:textColor="?android:attr/textColorPrimary" />

                <TextView android:id="@+id/am_pm"
                    android:layout_width="wrap_content"
```

```
                    android:layout_height="wrap_content"
                    android:textAppearance="?android:attr/textAppearanceSmall"
                    android:textStyle="bold"
                    android:textColor="?android:attr/textColorPrimary" />
            </LinearLayout>
      </com.android.superdeskclock.DigitalClock>
    </LinearLayout>
</LinearLayout>
```

设置主界面的对应程序文件是 AlarmClock.java，具体实现流程如下。

（1）定义函数 bindView()，根据 findViewById 获取系统内的闹钟，并将获取的数据绑定到 ListView 控件中，以列表样式显示出来。函数 bindView()的具体实现代码如下所示。

```
public void bindView(View view, Context context, Cursor cursor) {
   final Alarm alarm = new Alarm(cursor);

   View indicator = view.findViewById(R.id.indicator);

   // Set the initial resource for the bar image.
   final ImageView barOnOff = (ImageView) indicator.findViewById(R.id.bar_onoff);
   barOnOff.setImageResource(alarm.enabled ? R.drawable.ic_indicator_on : R.drawable.ic_indicator_off);

   // Set the initial state of the clock "checkbox"
   final CheckBox clockOnOff =(CheckBox) indicator.findViewById(R.id.clock_onoff);
   clockOnOff.setChecked(alarm.enabled);

   //新增
   this.context=context;
   // Clicking outside the "checkbox" should also change the state.
   indicator.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            myDialog = ProgressDialog.show(AlarmTimeAdapter.this.context,"提示","请稍候",true);
                new Thread(){
                    public void run(){
                        try{
                            sleep(800);
                        }catch (Exception e){
                            e.printStackTrace();
                        }finally{
                            // 卸载所创建的 myDialog 对象
                            myDialog.dismiss();
                        }
                    }
                }.start();
          try {
                Thread.sleep(500);
              } catch (InterruptedException e) {
                e.printStackTrace();
                              }
              clockOnOff.toggle();
              updateIndicatorAndAlarm(clockOnOff.isChecked(),barOnOff, alarm);
                 }
```

```
        });

        DigitalClock digitalClock =(DigitalClock) view.findViewById(R.id.digitalClock);

        // set the alarm text
        final Calendar c = Calendar.getInstance();
        c.set(Calendar.HOUR_OF_DAY, alarm.hour);
        c.set(Calendar.MINUTE, alarm.minutes);
        digitalClock.updateTime(c);
        digitalClock.setTypeface(Typeface.DEFAULT);

        // Set the repeat text or leave it blank if it does not repeat.
        TextView daysOfWeekView = (TextView) digitalClock.findViewById(R.id.daysOfWeek);
        final String daysOfWeekStr = alarm.daysOfWeek.toString(AlarmClock.this, false);
        if (daysOfWeekStr != null && daysOfWeekStr.length() != 0) {
            daysOfWeekView.setText(daysOfWeekStr);
            daysOfWeekView.setVisibility(View.VISIBLE);
        } else {
            daysOfWeekView.setVisibility(View.GONE);
        }

        // Display the label
        TextView labelView = (TextView) view.findViewById(R.id.label);
        if (alarm.label != null && alarm.label.length() != 0) {
            labelView.setText(alarm.label);
            labelView.setVisibility(View.VISIBLE);
        } else {
            labelView.setVisibility(View.GONE);
        }
    }
};
```

（2）定义函数 onContextItemSelected()，功能是根据用户选择确认列表中的闹钟是否处于可用状态，并根据用户选择来到不同状态的处理界面。函数 onContextItemSelected()的具体实现代码如下所示。

```
@Override
public boolean onContextItemSelected(final MenuItem item) {
    final AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
    final int id = (int) info.id;
    switch (item.getItemId()) {
        case R.id.delete_alarm:
            //确认闹钟已被删除
            new AlertDialog.Builder(this)
                    .setTitle(getString(R.string.delete_alarm))
                    .setMessage(getString(R.string.delete_alarm_confirm))
                    .setPositiveButton(android.R.string.ok,
                            new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface d, int w) {
                                    Alarms.deleteAlarm(AlarmClock.this, id);
                                }
                            })
                    .setNegativeButton(android.R.string.cancel, null)
```

```
                    .show();
            return true;

        case R.id.enable_alarm:
            final Cursor c = (Cursor) mAlarmsList.getAdapter().getItem(info.position);
            final Alarm alarm = new Alarm(c);
            //修改
            Alarms.enableAlarm(this, alarm.id, !alarm.enabled);
            if (!alarm.enabled) {
                SetAlarm.popAlarmSetToast(this, alarm.hour, alarm.minutes, alarm.daysOfWeek);
            }
            return true;

        case R.id.edit_alarm:
            Intent intent = new Intent(this, SetAlarm.class);
            intent.putExtra(Alarms.ALARM_ID, id);
            startActivity(intent);
            return true;

        default:
            break;
    }
    return super.onContextItemSelected(item);
}
```

（3）定义函数 onCreate()，功能是显示当前的界面。定义函数 updateLayout()，根据用户选择更新显示为对应的界面。具体实现代码如下所示。

```
@Override
protected void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    mFactory = LayoutInflater.from(this);
    mPrefs = getSharedPreferences(PREFERENCES, 0);
    mCursor = Alarms.getAlarmsCursor(getContentResolver());

    updateLayout();
}

private void updateLayout() {
    setContentView(R.layout.alarm_clock);
    mAlarmsList = (ListView) findViewById(R.id.alarms_list);
    AlarmTimeAdapter adapter = new AlarmTimeAdapter(this, mCursor);
    mAlarmsList.setAdapter(adapter);
    mAlarmsList.setVerticalScrollBarEnabled(true);
    mAlarmsList.setOnItemClickListener(this);
    mAlarmsList.setOnCreateContextMenuListener(this);

    View addAlarm = findViewById(R.id.add_alarm);
    addAlarm.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                addNewAlarm();
```

```
                }
            });
        // 当在屏幕中聚焦时选择这个选项
        addAlarm.setOnFocusChangeListener(new View.OnFocusChangeListener() {
                public void onFocusChange(View v, boolean hasFocus) {
                    v.setSelected(hasFocus);
                }
        });

        ImageButton deskClock =
                (ImageButton) findViewById(R.id.desk_clock_button);
        deskClock.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    startActivity(new Intent(AlarmClock.this, DeskClock.class));
                }
        });
}
```

（4）定义函数 addNewAlarm()，功能是启动添加闹钟界面。定义函数 onCreateContextMenu()，功能是创建一个上下文菜单。具体实现代码如下所示。

```
private void addNewAlarm() {
    startActivity(new Intent(this, SetAlarm.class));
}

@Override
protected void onDestroy() {
    super.onDestroy();
    ToastMaster.cancelToast();
    mCursor.deactivate();
}

@Override
public void onCreateContextMenu(ContextMenu menu, View view,
        ContextMenuInfo menuInfo) {
    //从 xml 文件创建菜单
    getMenuInflater().inflate(R.menu.context_menu, menu);

    //使用当前项目创建自定义视图的页眉
    final AdapterContextMenuInfo info = (AdapterContextMenuInfo) menuInfo;
    final Cursor c =
            (Cursor) mAlarmsList.getAdapter().getItem((int) info.position);
    final Alarm alarm = new Alarm(c);

    //构建日历计算时间
    final Calendar cal = Calendar.getInstance();
    cal.set(Calendar.HOUR_OF_DAY, alarm.hour);
    cal.set(Calendar.MINUTE, alarm.minutes);
    final String time = Alarms.formatTime(this, cal);

    //设置自定义视图和每个程序的文本
    final View v = mFactory.inflate(R.layout.context_menu_header, null);
```

```
    TextView textView = (TextView) v.findViewById(R.id.header_time);
    textView.setText(time);
    textView = (TextView) v.findViewById(R.id.header_label);
    textView.setText(alarm.label);

    //在菜单上设置自定义视图
    menu.setHeaderView(v);
    // Change the text based on the state of the alarm.
    if (alarm.enabled) {
        menu.findItem(R.id.enable_alarm).setTitle(R.string.disable_alarm);
    }
}
```

当按下 MENU 手机按键后会在屏幕底部弹出这个菜单项，如图 15-4 所示。

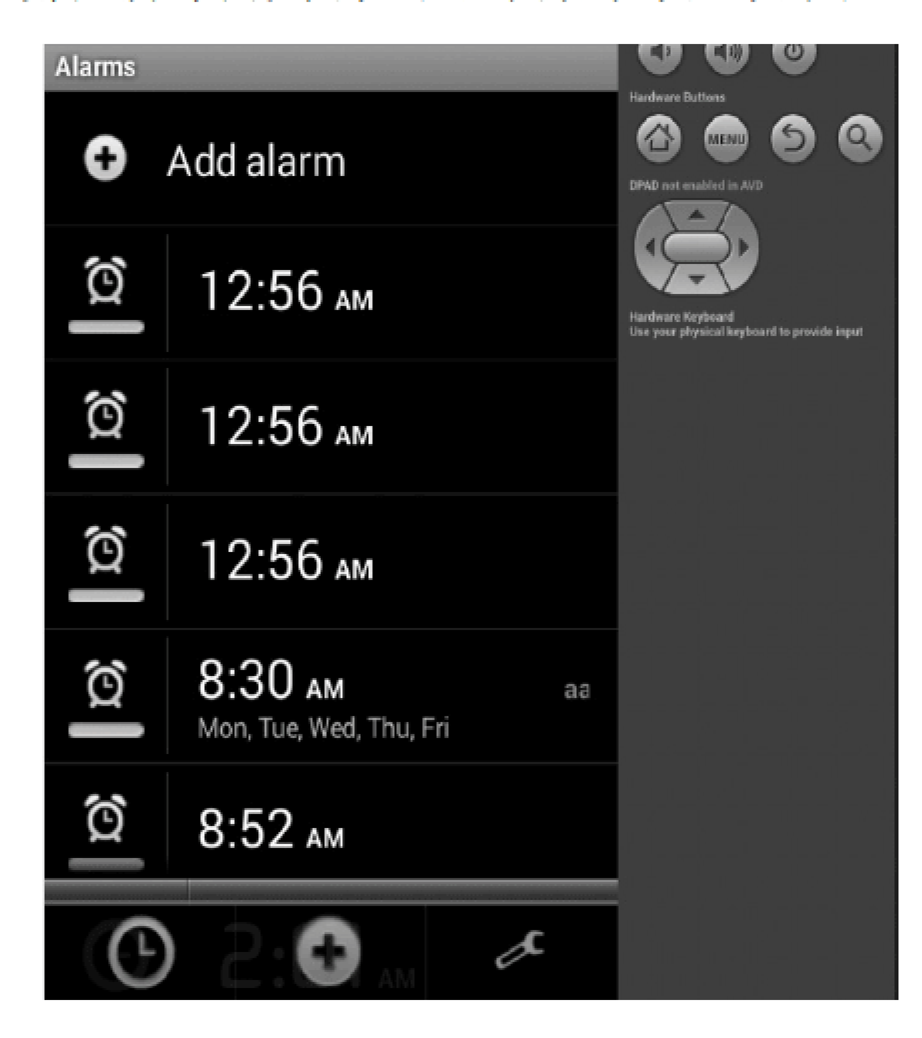


图 15-4　屏幕底部的上下文菜单项

在图 15-4 所示的菜单项中，包含了闹钟设置、添加闹钟和闹钟主界面 3 个选项。

（5）定义函数 onOptionsItemSelected()，功能是根据用户在函数 onCreateContextMenu()中创建的上下文菜单上选择的项，来到不同的界面。具体实现代码如下所示。

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item_settings:
            startActivity(new Intent(this, SettingsActivity.class));
            return true;
        case R.id.menu_item_desk_clock:
            startActivity(new Intent(this, DeskClock.class));
            return true;
        case R.id.menu_item_add_alarm:
            addNewAlarm();
            return true;
        default:
            break;
    }
    return super.onOptionsItemSelected(item);
```

```
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.alarm_list_menu, menu);
    return super.onCreateOptionsMenu(menu);
}

public void onItemClick(AdapterView parent, View v, int pos, long id) {
    Intent intent = new Intent(this, SetAlarm.class);
    intent.putExtra(Alarms.ALARM_ID, (int) id);
    startActivity(intent);
}
```

15.3.2 设置闹钟界面

当在图 15-3 所示的闹钟列表界面中选择一个闹钟时，会来到设置闹钟界面。在此界面中可以设置被选中闹钟的选项，例如开关、时间、重复次数、铃声和震动等。设置闹钟界面的执行效果如图 15-5 所示。

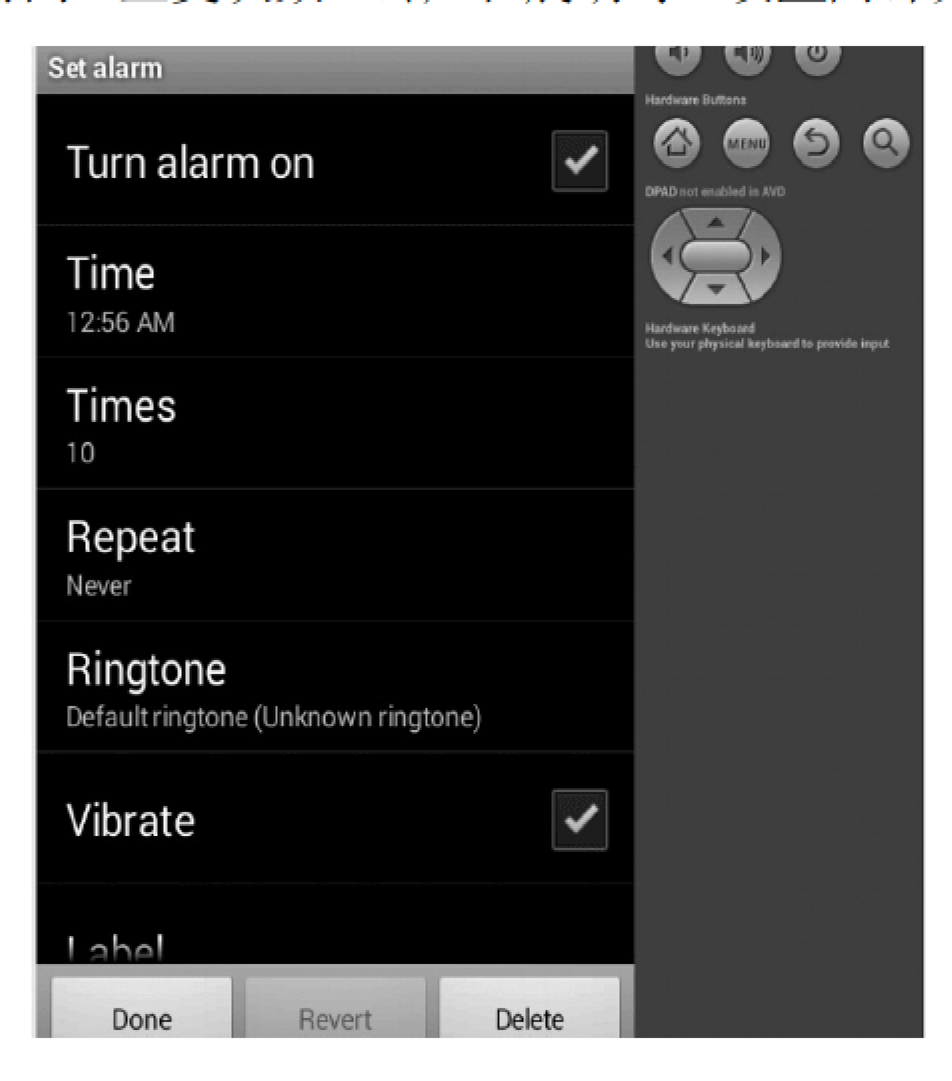


图 15-5 设置闹钟界面的执行效果

图 15-5 所示的界面效果是由文件 set_alarm.xml 实现布局的，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <ListView android:id="@android:id/list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:drawSelectorOnTop="false"/>
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
```

```
        style="@android:style/ButtonBar">
        <Button android:id="@+id/alarm_save"
            android:focusable="true"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:text="@string/done"/>
        <Button android:id="@+id/alarm_revert"
            android:focusable="true"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:text="@string/revert"/>
        <Button android:id="@+id/alarm_delete"
            android:focusable="true"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:text="@string/delete"/>
    </LinearLayout>
</LinearLayout>
```

设置闹钟界面的程序文件是 SetAlarm.java，具体实现流程如下所示。

（1）定义继承于 PreferenceActivity 的类 SetAlarm，用于管理系统中的每一个闹钟。具体实现代码如下所示。

```
public class SetAlarm extends PreferenceActivity
        implements TimePickerDialog.OnTimeSetListener,
        Preference.OnPreferenceChangeListener {

    private EditTextPreference mLabel;

    //新增
    private EditTextPreference times;
    private EditTextPreference interval;


    private CheckBoxPreference mEnabledPref;
    private Preference mTimePref;
    private SetBellPreference mAlarmPref;
    private RadioButton RadioButton;
    private CheckBoxPreference mVibratePref;
    private RepeatPreference mRepeatPref;
    private MenuItem mTestAlarmItem;

    private int     mId;
    private int     mHour;
    private int     mMinutes;
    private boolean mTimePickerCancelled;
    private Alarm   mOriginalAlarm;
```

（2）定义函数 onCreate 以创建此界面的显示视图，加载显示系统中原来对这个闹钟的设置。函数

onCreate()的具体实现代码如下所示。

```
/**
 *设置一个 Alarm 闹钟，通过外部 Alarms.ALARM_ID 实现
 *诸如其他 Activity 之类的 Alarm 对象
 */
@Override
protected void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    // Override the default content view.
    setContentView(R.layout.set_alarm);
    addPreferencesFromResource(R.xml.alarm_prefs);
    //新增
    times = (EditTextPreference) findPreference("times");
    times.setOnPreferenceChangeListener(
            new Preference.OnPreferenceChangeListener() {
                public boolean onPreferenceChange(Preference p,
                        Object newValue) {
                    String val = (String) newValue;
                    // Set the summary based on the new label.
                    p.setSummary(val);
                    if (val != null && !val.equals(times.getText())) {
                        // Call through to the generic listener.
                        return SetAlarm.this.onPreferenceChange(p,
                            newValue);
                    }
                    return true;
                }
            });
    // 检索值，获取每一个偏好值
    mLabel = (EditTextPreference) findPreference("label");
    mLabel.setOnPreferenceChangeListener(
            new Preference.OnPreferenceChangeListener() {
                public boolean onPreferenceChange(Preference p,
                        Object newValue) {
                    String val = (String) newValue;
                    // Set the summary based on the new label
                    p.setSummary(val);
                    if (val != null && !val.equals(mLabel.getText())) {
                        // Call through to the generic listener
                        return SetAlarm.this.onPreferenceChange(p,
                            newValue);
                    }
                    return true;
                }
            });
    mEnabledPref = (CheckBoxPreference) findPreference("enabled");
    mEnabledPref.setOnPreferenceChangeListener(
            new Preference.OnPreferenceChangeListener() {
                public boolean onPreferenceChange(Preference p,
```

```
                    Object newValue) {
                // Pop a toast when enabling alarms
                if (!mEnabledPref.isChecked()) {
                    popAlarmSetToast(SetAlarm.this, mHour, mMinutes,
                        mRepeatPref.getDaysOfWeek());
                }
                return SetAlarm.this.onPreferenceChange(p, newValue);
            }
        });
mTimePref = findPreference("time");
mAlarmPref = (SetBellPreference) findPreference("alarm");
mAlarmPref.setOnPreferenceChangeListener(this);
mVibratePref = (CheckBoxPreference) findPreference("vibrate");
mVibratePref.setOnPreferenceChangeListener(this);
mRepeatPref = (RepeatPreference) findPreference("setRepeat");
mRepeatPref.setOnPreferenceChangeListener(this);

Intent i = getIntent();
mId = i.getIntExtra(Alarms.ALARM_ID, -1);
if (Log.LOGV) {
    Log.v("In SetAlarm, alarm id = " + mId);
}

Alarm alarm = null;
if (mId == -1) {
    //没有闹钟则新建一个
    alarm = new Alarm();
} else {
    /* 从数据库中显示这个闹钟的详细信息 */
    alarm = Alarms.getAlarm(getContentResolver(), mId);
    // Bad alarm, bail to avoid a NPE.
    if (alarm == null) {
        finish();
        return;
    }
}
mOriginalAlarm = alarm;

updatePrefs(mOriginalAlarm);

//选中时高亮显示
getListView().setItemsCanFocus(true);

//给每个按钮附加操作
Button b = (Button) findViewById(R.id.alarm_save);
b.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            saveAlarm();
            finish();
        }
});
```

```
        final Button revert = (Button) findViewById(R.id.alarm_revert);
        revert.setEnabled(false);
        revert.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    int newId = mId;
                    updatePrefs(mOriginalAlarm);
                    // "Revert" on a newly created alarm should delete it
                    if (mOriginalAlarm.id == -1) {
                        Alarms.deleteAlarm(SetAlarm.this, newId);
                    } else {
                        saveAlarm();
                    }
                    revert.setEnabled(false);
                }
        });
        b = (Button) findViewById(R.id.alarm_delete);
        if (mId == -1) {
            b.setEnabled(false);
        } else {
            b.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    deleteAlarm();
                }
            });
        }

        //如果改变了时间，则弹出时间选择器
        if (mId == -1) {
            // Assume the user hit cancel
            mTimePickerCancelled = true;
            showTimePicker();
        }
    }
    private static final Handler sHandler = new Handler();
```

（3）定义函数 onPreferenceChange()以处理偏好变化，具体实现代码如下所示。

```
public boolean onPreferenceChange(final Preference p, Object newValue) {
    //定义异步方法保存偏好值的更改
    sHandler.post(new Runnable() {
        public void run() {
            //编辑可设置的偏好
            if (p != mEnabledPref) {
                mEnabledPref.setChecked(true);
            }
            saveAlarmAndEnableRevert();
        }
    });
    return true;
}
```

（4）定义函数 updatePrefs()以更新对这个闹钟的设置，具体实现代码如下所示。

```
private void updatePrefs(Alarm alarm) {
```

```
    mId = alarm.id;
    mEnabledPref.setChecked(alarm.enabled);
    mLabel.setText(alarm.label);
    mLabel.setSummary(alarm.label);
    mHour = alarm.hour;
    mMinutes = alarm.minutes;
    mRepeatPref.setDaysOfWeek(alarm.daysOfWeek);
    mVibratePref.setChecked(alarm.vibrate);
    // Give the alert uri to the preference
    mAlarmPref.setAlert(alarm.alert);

    //新增
    times.setText(""+alarm.times);
    times.setSummary(""+alarm.times);


    updateTime();
}
```

（5）定义函数 onTimeSet()以设置新闹钟的时间，定义函数 upTimeSet()更改这个闹钟的时间。具体实现代码如下所示。

```
public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
    // onTimeSet is called when the user clicks "Set"
    mTimePickerCancelled = false;
    mHour = hourOfDay;
    mMinutes = minute;
    updateTime();
    // If the time has been changed, enable the alarm.
    mEnabledPref.setChecked(true);
    // Save the alarm and pop a toast.
    popAlarmSetToast(this, saveAlarmAndEnableRevert());
}
private void updateTime() {
    if (Log.LOGV) {
        Log.v("updateTime " + mId);
    }
    mTimePref.setSummary(Alarms.formatTime(this, mHour, mMinutes,
            mRepeatPref.getDaysOfWeek()));
}
```

（6）定义函数 saveAlarmAndEnableRevert()以保存闹钟，并设置 Revert 按钮不可用。具体实现代码如下所示。

```
private long saveAlarmAndEnableRevert() {
    // Enable "Revert" to go back to the original Alarm
    final Button revert = (Button) findViewById(R.id.alarm_revert);
    revert.setEnabled(true);
    return saveAlarm();
}
```

（7）定义函数 saveAlarm()以保存对当前闹钟的设置，具体实现代码如下所示。

```
private long saveAlarm() {
    Alarm alarm = new Alarm();
```

```
        alarm.id = mId;
        alarm.enabled = mEnabledPref.isChecked();
        alarm.hour = mHour;
        alarm.minutes = mMinutes;
        alarm.daysOfWeek = mRepeatPref.getDaysOfWeek();
        alarm.vibrate = mVibratePref.isChecked();
        alarm.label = mLabel.getText();
        alarm.alert = mAlarmPref.getAlert();

        //新增
        String timesText=times.getText();
        alarm.times=Integer.parseInt(timesText==null||"".equals(timesText)?"0":timesText);
        alarm.interval=0;

        long time;
        if (alarm.id == -1) {
            time = Alarms.addAlarm(this, alarm);
            // addAlarm populates the alarm with the new id. Update mId so that
            // changes to other preferences update the new alarm
            mId = alarm.id;
        } else {
            time = Alarms.setAlarm(this, alarm);
        }
        return time;
}
```

（8）定义函数 deleteAlarm()以删除当前闹钟，在单击屏幕底部的 Delete 按钮时被触发。函数 deleteAlarm()的具体实现代码如下所示。

```
private void deleteAlarm() {
        new AlertDialog.Builder(this)
                .setTitle(getString(R.string.delete_alarm))
                .setMessage(getString(R.string.delete_alarm_confirm))
                .setPositiveButton(android.R.string.ok,
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface d, int w) {
                                Alarms.deleteAlarm(SetAlarm.this, mId);
                                finish();
                            }
                        })
                .setNegativeButton(android.R.string.cancel, null)
                .show();
}
```

（9）定义函数 popAlarmSetToast()，功能是当设置完毕后单击屏幕底部的 Done 按钮后显示一个闹钟设置提醒信息，如图 15-6 所示。

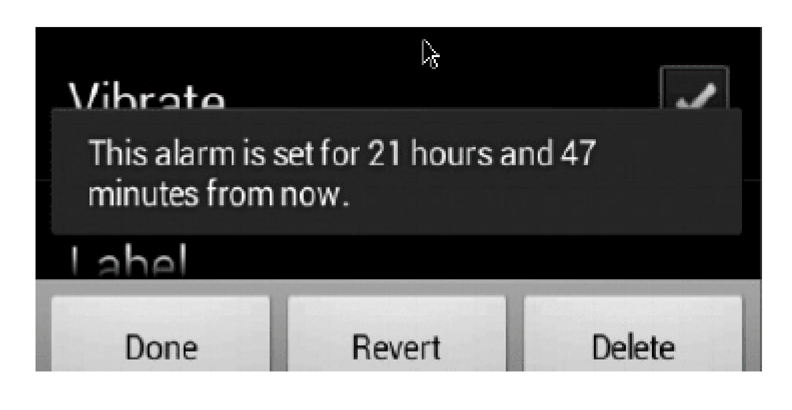
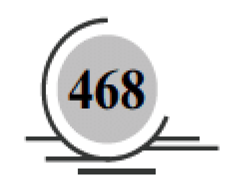


图 15-6　提醒信息效果



函数 popAlarmSetToast()的具体实现代码如下所示。

```
static void popAlarmSetToast(Context context, int hour, int minute,
                                        Alarm.DaysOfWeek daysOfWeek) {
    popAlarmSetToast(context,
            Alarms.calculateAlarm(hour, minute, daysOfWeek)
            .getTimeInMillis());
}

private static void popAlarmSetToast(Context context, long timeInMillis) {
    String toastText = formatToast(context, timeInMillis);
    Toast toast = Toast.makeText(context, toastText, Toast.LENGTH_LONG);
    ToastMaster.setToast(toast);
    toast.show();
}
```

（10）定义函数 formatToast()，功能是设置提醒信息的内容，具体实现代码如下所示。

```
static String formatToast(Context context, long timeInMillis) {
    long delta = timeInMillis - System.currentTimeMillis();
    long hours = delta / (1000 * 60 * 60);
    long minutes = delta / (1000 * 60) % 60;
    long days = hours / 24;
    hours = hours % 24;

    String daySeq = (days == 0) ? "" :
            (days == 1) ? context.getString(R.string.day) :
            context.getString(R.string.days, Long.toString(days));

    String minSeq = (minutes == 0) ? "" :
            (minutes == 1) ? context.getString(R.string.minute) :
            context.getString(R.string.minutes, Long.toString(minutes));

    String hourSeq = (hours == 0) ? "" :
            (hours == 1) ? context.getString(R.string.hour) :
            context.getString(R.string.hours, Long.toString(hours));

    boolean dispDays = days > 0;
    boolean dispHour = hours > 0;
    boolean dispMinute = minutes > 0;

    int index = (dispDays ? 1 : 0) |
                (dispHour ? 2 : 0) |
                (dispMinute ? 4 : 0);

    String[ ] formats = context.getResources().getStringArray(R.array.alarm_set);
    return String.format(formats[index], daySeq, hourSeq, minSeq);
}
 public void setInterval(EditTextPreference interval) {
      this.interval = interval;
 }
```

15.3.3 闹钟提醒模块

在本实例中，涉及了多处闹钟提醒模块，例如闹钟到时响起时的提醒、闹钟全屏提醒。在本节的内容中，将详细讲解闹钟提醒模块的具体实现过程。

（1）界面布局文件是 alarm_alert.xml，具体实现代码如下所示。

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center_horizontal"
        android:background="@drawable/dialog"
        android:orientation="vertical">
        <TextView android:id="@+id/alertTitle"
            style="?android:attr/textAppearanceLarge"
            android:padding="5dip"
            android:singleLine="true"
            android:ellipsize="end"
            android:gravity="center"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <ImageView
            android:layout_width="fill_parent"
            android:layout_height="1dip"
            android:scaleType="fitXY"
            android:gravity="fill_horizontal"
            android:src="@drawable/dialog_divider_horizontal_light"
            android:layout_marginLeft="10dip"
            android:layout_marginRight="10dip"/>
        <com.android.superdeskclock.DigitalClock
            style="@style/clock"
            android:paddingTop="30dip"
            android:paddingBottom="30dip"
            android:baselineAligned="true"
            android:gravity="center_horizontal">
            <TextView android:id="@+id/timeDisplay"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="64sp"
                android:textColor="?android:attr/textColorPrimary"/>
            <TextView android:id="@+id/am_pm"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textStyle="bold"
```

```
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:textColor="?android:attr/textColorPrimary"/>
        </com.android.superdeskclock.DigitalClock>
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            style="@android:style/ButtonBar">
            <Button
                android:id="@+id/snooze"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="3"
                android:text="@string/alarm_alert_snooze_text" />
            <!-- blank stretchable view -->
            <View
                android:layout_width="2dip"
                android:layout_height="2dip"
                android:layout_gravity="fill_horizontal"
                android:layout_weight="1"/>
            <Button
                android:id="@+id/dismiss"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="3"
                android:text="@string/alarm_alert_dismiss_text" />
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

（2）再看文件 AlarmAlert.java，功能是实现全屏幕的闹钟提示，使用持久性的可见指示器和指定音乐实现。文件 AlarmAlert.java 的具体实现代码如下所示。

```
public class AlarmAlert extends AlarmAlertFullScreen {
    //如果尝试操作键盘锁的 5 次以上，则退出全屏幕活动
    private int mKeyguardRetryCount;
    private final int MAX_KEYGUARD_CHECKS = 5;

    private final Handler mHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            handleScreenOff((KeyguardManager) msg.obj);
        }
    };

    private final BroadcastReceiver mScreenOffReceiver =
            new BroadcastReceiver() {
                @Override
                public void onReceive(Context context, Intent intent) {
                    KeyguardManager km =
                            (KeyguardManager) context.getSystemService(
                            Context.KEYGUARD_SERVICE);
                    handleScreenOff(km);
```

```
                }
            };

    @Override
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        //当屏幕恢复时监听屏幕关闭，这样用户不需要解锁解除闹钟
        registerReceiver(mScreenOffReceiver,
                new IntentFilter(Intent.ACTION_SCREEN_OFF));
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        unregisterReceiver(mScreenOffReceiver);
        //删除任何键盘锁消息
        mHandler.removeMessages(0);
    }

    @Override
    public void onBackPressed() {
        finish();
    }

    private boolean checkRetryCount() {
        if (mKeyguardRetryCount++ >= MAX_KEYGUARD_CHECKS) {
            Log.e("Tried to read keyguard status too many times, bailing...");
            return false;
        }
        return true;
    }
    private void handleScreenOff(final KeyguardManager km) {
        if (!km.inKeyguardRestrictedInputMode() && checkRetryCount()) {
            if (checkRetryCount()) {
                mHandler.sendMessageDelayed(mHandler.obtainMessage(0, km), 500);
            }
        } else {
            //启动全屏活动但不打开屏幕
            Intent i = new Intent(this, AlarmAlertFullScreen.class);
            i.putExtra(Alarms.ALARM_INTENT_EXTRA, mAlarm);
            i.putExtra(SCREEN_OFF, true);
            startActivity(i);
            finish();
        }
    }
}
```

（3）再看文件 AlarmAlertFullScreen.java，功能是实现有背景桌面的、被锁定的全屏幕闹钟提示效果，此类型闹钟使用持久性的可见指示器和指定音乐实现。文件 AlarmAlertFullScreen.java 的具体实现代码如下所示。

```
public class AlarmAlertFullScreen extends Activity {

    //下面的值和文件“xml/settings.xml”中的相对应
    private static final String DEFAULT_SNOOZE = "10";
    private static final String DEFAULT_VOLUME_BEHAVIOR = "2";
    protected static final String SCREEN_OFF = "screen_off";

    protected Alarm mAlarm;
    private int mVolumeBehavior;

    //从 AlarmKlaxon 接收 ALARM_KILLED 操作，也从其他程序接收 ALARM_SNOOZE_ACTION / ALARM_
DISMISS_ACTION
    private BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if (action.equals(Alarms.ALARM_SNOOZE_ACTION)) {
                snooze();
            } else if (action.equals(Alarms.ALARM_DISMISS_ACTION)) {
                dismiss(false);
            } else {
                Alarm alarm = intent.getParcelableExtra(Alarms.ALARM_INTENT_EXTRA);
                if (alarm != null && mAlarm.id == alarm.id) {
                    dismiss(true);
                }
            }
        }
    };

    @Override
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        mAlarm = getIntent().getParcelableExtra(Alarms.ALARM_INTENT_EXTRA);

        //获取声音和相机设置
        final String vol =
                PreferenceManager.getDefaultSharedPreferences(this)
                .getString(SettingsActivity.KEY_VOLUME_BEHAVIOR,
                        DEFAULT_VOLUME_BEHAVIOR);
        mVolumeBehavior = Integer.parseInt(vol);

        requestWindowFeature(android.view.Window.FEATURE_NO_TITLE);

        final Window win = getWindow();
        win.addFlags(WindowManager.LayoutParams.FLAG_SHOW_WHEN_LOCKED
                | WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD);
        //打开屏幕
        if (!getIntent().getBooleanExtra(SCREEN_OFF, false)) {
            win.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON
                    | WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON
```

```
//                          | WindowManager.LayoutParams.FLAG_ALLOW_LOCK_WHILE_SCREEN_ON
                          );
        }

        updateLayout();

        //3 个闹钟按钮的处理：关闭/睡眠/解散
        IntentFilter filter = new IntentFilter(Alarms.ALARM_KILLED);
        filter.addAction(Alarms.ALARM_SNOOZE_ACTION);
        filter.addAction(Alarms.ALARM_DISMISS_ACTION);
        registerReceiver(mReceiver, filter);
    }

    private void setTitle() {
        String label = mAlarm.getLabelOrDefault(this);
        TextView title = (TextView) findViewById(R.id.alertTitle);
        title.setText(label);
    }

    private void updateLayout() {
        LayoutInflater inflater = LayoutInflater.from(this);

        setContentView(inflater.inflate(R.layout.alarm_alert, null));

        /* snooze behavior: pop a snooze confirmation view, kick alarm
           manager. */
        Button snooze = (Button) findViewById(R.id.snooze);
        snooze.requestFocus();
        snooze.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                snooze();
            }
        });

        /* dismiss 按钮：关闭提醒*/
        findViewById(R.id.dismiss).setOnClickListener(
                new Button.OnClickListener() {
                    public void onClick(View v) {
                        dismiss(false);
                    }
                });

        /*在闹钟设置标题 */
        setTitle();
    }

    //暂停此警报
    private void snooze() {
        //不要贪睡，snooze 按钮被禁用
        if (!findViewById(R.id.snooze).isEnabled()) {
            dismiss(false);
```

```
            return;
        }
        final String snooze =
                PreferenceManager.getDefaultSharedPreferences(this)
                .getString(SettingsActivity.KEY_ALARM_SNOOZE, DEFAULT_SNOOZE);
        int snoozeMinutes = Integer.parseInt(snooze);

        final long snoozeTime = System.currentTimeMillis()
                + (1000 * 60 * snoozeMinutes);
        Alarms.saveSnoozeAlert(AlarmAlertFullScreen.this, mAlarm.id,
                snoozeTime);

        //根据 snooze 和 update 的设置得到显示时间
        final Calendar c = Calendar.getInstance();
        c.setTimeInMillis(snoozeTime);

        // Append (snoozed) to the label.
        String label = mAlarm.getLabelOrDefault(this);
        label = getString(R.string.alarm_notify_snooze_label, label);

        //通知用户，闹钟被暂停
        Intent cancelSnooze = new Intent(this, AlarmReceiver.class);
        cancelSnooze.setAction(Alarms.CANCEL_SNOOZE);
        cancelSnooze.putExtra(Alarms.ALARM_ID, mAlarm.id);
        PendingIntent broadcast =
                PendingIntent.getBroadcast(this, mAlarm.id, cancelSnooze, 0);
        NotificationManager nm = getNotificationManager();
        Notification n = new Notification(R.drawable.stat_notify_alarm,
                label, 0);
        n.setLatestEventInfo(this, label,
                getString(R.string.alarm_notify_snooze_text,
                    Alarms.formatTime(this, c)), broadcast);
        n.flags |= Notification.FLAG_AUTO_CANCEL
                | Notification.FLAG_ONGOING_EVENT;
        nm.notify(mAlarm.id, n);

        String displayTime = getString(R.string.alarm_alert_snooze_set,
                snoozeMinutes);
        //故意延迟小睡的时间
        Log.v(displayTime);

        //在提醒中显示小睡时间
        Toast.makeText(AlarmAlertFullScreen.this, displayTime,
                Toast.LENGTH_LONG).show();
        stopService(new Intent(Alarms.ALARM_ALERT_ACTION));
        finish();
    }

    private NotificationManager getNotificationManager() {
        return (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    }
```

```
// Dismiss 闹钟
private void dismiss(boolean killed) {
    Log.i(killed ? "Alarm killed" : "Alarm dismissed by user");
    // The service told us that the alarm has been killed, do not modify
    // the notification or stop the service
    if (!killed) {
        // Cancel the notification and stop playing the alarm
        NotificationManager nm = getNotificationManager();
        nm.cancel(mAlarm.id);
        stopService(new Intent(Alarms.ALARM_ALERT_ACTION));
    }
    finish();
}
@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    if (Log.LOGV) Log.v("AlarmAlert.OnNewIntent()");
    mAlarm = intent.getParcelableExtra(Alarms.ALARM_INTENT_EXTRA);
    setTitle();
}
@Override
protected void onResume() {
    super.onResume();
    // If the alarm was deleted at some point, disable snooze
    if (Alarms.getAlarm(getContentResolver(), mAlarm.id) == null) {
        Button snooze = (Button) findViewById(R.id.snooze);
        snooze.setEnabled(false);
    }
}

@Override
public void onDestroy() {
    super.onDestroy();
    if (Log.LOGV) Log.v("AlarmAlert.onDestroy()");
    // No longer care about the alarm being killed
    unregisterReceiver(mReceiver);
}

@Override
public boolean dispatchKeyEvent(KeyEvent event) {
    // Do this on key down to handle a few of the system keys
    boolean up = event.getAction() == KeyEvent.ACTION_UP;
    switch (event.getKeyCode()) {
        // Volume keys and camera keys dismiss the alarm
        case KeyEvent.KEYCODE_VOLUME_UP:
        case KeyEvent.KEYCODE_VOLUME_DOWN:
        case KeyEvent.KEYCODE_CAMERA:
        case KeyEvent.KEYCODE_FOCUS:
            if (up) {
                switch (mVolumeBehavior) {
```

```
                    case 1:
                        snooze();
                        break;

                    case 2:
                        dismiss(false);
                        break;

                    default:
                        break;
                }
            }
            return true;
        default:
            break;
    }
    return super.dispatchKeyEvent(event);
}
```

15.3.4　重复设置

在系统中设置一个闹钟时，可以设置这个闹钟在其他时间也起作用，例如每个周一或周二。此功能是通过文件 RepeatPreference.java 实现的，具体实现代码如下所示。

```
public class RepeatPreference extends ListPreference {
    // Initial value that can be set with the values saved in the database.
    private Alarm.DaysOfWeek mDaysOfWeek = new Alarm.DaysOfWeek(0);
    // New value that will be set if a positive result comes back from the
    // dialog
    private Alarm.DaysOfWeek mNewDaysOfWeek = new Alarm.DaysOfWeek(0);

    public RepeatPreference(Context context, AttributeSet attrs) {
        super(context, attrs);

        String[ ] weekdays = new DateFormatSymbols().getWeekdays();
        String[ ] values = new String[ ] {
            weekdays[Calendar.MONDAY],
            weekdays[Calendar.TUESDAY],
            weekdays[Calendar.WEDNESDAY],
            weekdays[Calendar.THURSDAY],
            weekdays[Calendar.FRIDAY],
            weekdays[Calendar.SATURDAY],
            weekdays[Calendar.SUNDAY],
        };
        setEntries(values);
        setEntryValues(values);
    }

    @Override
    protected void onDialogClosed(boolean positiveResult) {
        if (positiveResult) {
```

```
                mDaysOfWeek.set(mNewDaysOfWeek);
                setSummary(mDaysOfWeek.toString(getContext(), true));
                callChangeListener(mDaysOfWeek);
            }
        }

        @Override
        protected void onPrepareDialogBuilder(Builder builder) {
            CharSequence[ ] entries = getEntries();
            @SuppressWarnings("unused")
              CharSequence[ ] entryValues = getEntryValues();


            builder.setMultiChoiceItems(
                    entries, mDaysOfWeek.getBooleanArray(),
                    new DialogInterface.OnMultiChoiceClickListener() {
                        public void onClick(DialogInterface dialog, int which,
                                boolean isChecked) {
                            mNewDaysOfWeek.set(which, isChecked);
                        }
                    });
        }
        public void setDaysOfWeek(Alarm.DaysOfWeek dow) {
            mDaysOfWeek.set(dow);
            mNewDaysOfWeek.set(dow);
            setSummary(dow.toString(getContext(), true));
        }
        public Alarm.DaysOfWeek getDaysOfWeek() {
            return mDaysOfWeek;
        }
}
```

重复设置界面的执行效果如图 15-7 所示。

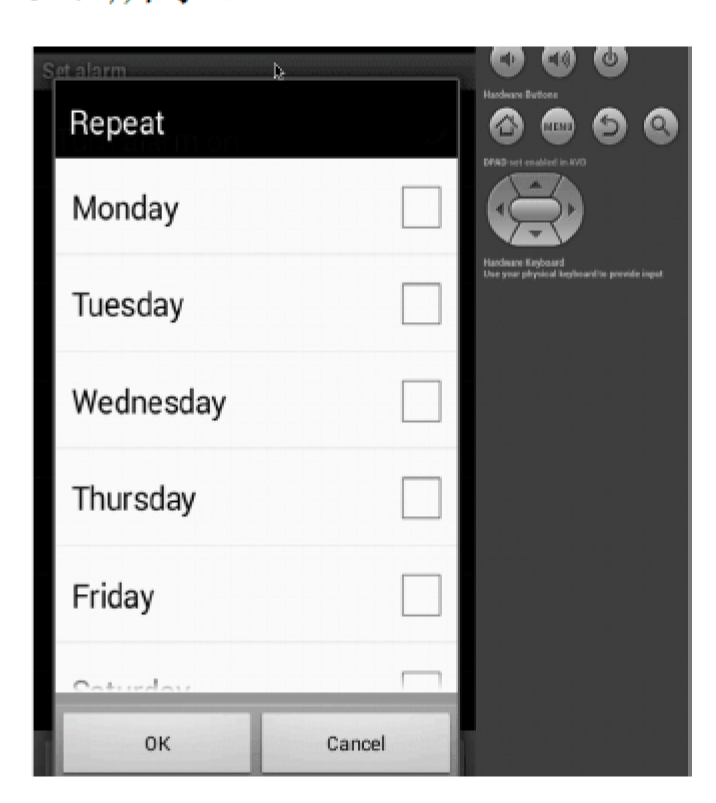
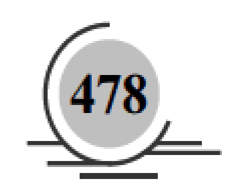


图 15-7 重复设置界面的执行效果

15.3.5 闹钟数据操作

在本系统中，采用了 SQLiteDatabase 数据库来保存系统中的数据，例如每个闹钟的时间、重复、



铃声和震动等信息。和数据操作相关的程序文件是 AlarmProvider.java，具体实现代码如下所示。

```
public class AlarmProvider extends ContentProvider {
    private SQLiteOpenHelper mOpenHelper;

    private static final int ALARMS = 1;
    private static final int ALARMS_ID = 2;
    private static final UriMatcher sURLMatcher = new UriMatcher(
            UriMatcher.NO_MATCH);

    static {
        sURLMatcher.addURI("com.android.superdeskclock", "alarm", ALARMS);
        sURLMatcher.addURI("com.android.superdeskclock", "alarm/#", ALARMS_ID);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper {
        private static final String DATABASE_NAME = "alarms.db";
        private static final int DATABASE_VERSION = 5;

        public DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
//创建保存数据库中的每一个选项
        @Override
        public void onCreate(SQLiteDatabase db) {
            db.execSQL("CREATE TABLE alarms (" +
                       "_id INTEGER PRIMARY KEY," +
                       "hour INTEGER, " +
                       "minutes INTEGER, " +
                       "daysofweek INTEGER, " +
                       "alarmtime INTEGER, " +
                       "enabled INTEGER, " +
                       "vibrate INTEGER, " +
                       "message TEXT, " +
                       "alert TEXT, " +
                       "times INTEGER, " +
                       "interval INTEGER);");

            //插入闹钟数据
            String insertMe = "INSERT INTO alarms " +
                    "(hour, minutes, daysofweek, alarmtime, enabled, vibrate, message, alert,times,interval) "
+
                    "VALUES ";
            db.execSQL(insertMe + "(8, 30, 31, 0, 0, 1, '', '',10,0);");
            db.execSQL(insertMe + "(9, 00, 96, 0, 0, 1, '', '',10,0);");
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int currentVersion) {
            if (Log.LOGV) Log.v(
                    "Upgrading alarms database from version " +
```

```
                    oldVersion + " to " + currentVersion +
                    ", which will destroy all old data");
            db.execSQL("DROP TABLE IF EXISTS alarms");
            onCreate(db);
        }
    }

    public AlarmProvider() {
    }

    @Override
    public boolean onCreate() {
        mOpenHelper = new DatabaseHelper(getContext());
        return true;
    }
//查询系统中的闹钟数据
    @Override
    public Cursor query(Uri url, String[ ] projectionIn, String selection,
            String[ ] selectionArgs, String sort) {
        SQLiteQueryBuilder qb = new SQLiteQueryBuilder();

        // Generate the body of the query
        int match = sURLMatcher.match(url);
        switch (match) {
            case ALARMS:
                qb.setTables("alarms");
                break;
            case ALARMS_ID:
                qb.setTables("alarms");
                qb.appendWhere("_id=");
                qb.appendWhere(url.getPathSegments().get(1));
                break;
            default:
                throw new IllegalArgumentException("Unknown URL " + url);
        }

        SQLiteDatabase db = mOpenHelper.getReadableDatabase();
        Cursor ret = qb.query(db, projectionIn, selection, selectionArgs,
                              null, null, sort);

        if (ret == null) {
            if (Log.LOGV) Log.v("Alarms.query: failed");
        } else {
            ret.setNotificationUri(getContext().getContentResolver(), url);
        }

        return ret;
    }
//获取类型信息
    @Override
    public String getType(Uri url) {
```

```
        int match = sURLMatcher.match(url);
        switch (match) {
            case ALARMS:
                return "vnd.android.cursor.dir/alarms";
            case ALARMS_ID:
                return "vnd.android.cursor.item/alarms";
            default:
                throw new IllegalArgumentException("Unknown URL");
        }
    }
//更新系统中的闹钟信息
    @Override
    public int update(Uri url, ContentValues values, String where, String[ ] whereArgs) {
        int count;
        long rowId = 0;
        int match = sURLMatcher.match(url);
        SQLiteDatabase db = mOpenHelper.getWritableDatabase();
        switch (match) {
            case ALARMS_ID: {
                String segment = url.getPathSegments().get(1);
                rowId = Long.parseLong(segment);
                count = db.update("alarms", values, "_id=" + rowId, null);
                break;
            }
            default: {
                throw new UnsupportedOperationException(
                        "Cannot update URL: " + url);
            }
        }
        if (Log.LOGV) Log.v("*** notifyChange() rowId: " + rowId + " url " + url);
        getContext().getContentResolver().notifyChange(url, null);
        return count;
    }

    @Override
    public Uri insert(Uri url, ContentValues initialValues) {
        if (sURLMatcher.match(url) != ALARMS) {
            throw new IllegalArgumentException("Cannot insert into URL: " + url);
        }

        ContentValues values = new ContentValues(initialValues);

        SQLiteDatabase db = mOpenHelper.getWritableDatabase();
        long rowId = db.insert("alarms", Alarm.Columns.MESSAGE, values);
        if (rowId < 0) {
            throw new SQLException("Failed to insert row into " + url);
        }
        if (Log.LOGV) Log.v("Added alarm rowId = " + rowId);

        Uri newUrl = ContentUris.withAppendedId(Alarm.Columns.CONTENT_URI, rowId);
        getContext().getContentResolver().notifyChange(newUrl, null);
```

```
            return newUrl;
        }
//删除信息
        public int delete(Uri url, String where, String[ ] whereArgs) {
            SQLiteDatabase db = mOpenHelper.getWritableDatabase();
            int count;
            long rowId = 0;
            switch (sURLMatcher.match(url)) {
                case ALARMS:
                    count = db.delete("alarms", where, whereArgs);
                    break;
                case ALARMS_ID:
                    String segment = url.getPathSegments().get(1);
                    rowId = Long.parseLong(segment);
                    if (TextUtils.isEmpty(where)) {
                        where = "_id=" + segment;
                    } else {
                        where = "_id=" + segment + " AND (" + where + ")";
                    }
                    count = db.delete("alarms", where, whereArgs);
                    break;
                default:
                    throw new IllegalArgumentException("Cannot delete from URL: " + url);
            }
            getContext().getContentResolver().notifyChange(url, null);
            return count;
        }
}
```

15.4 选择铃声音乐

知识点讲解：光盘:视频\知识点\第 15 章\选择铃声音乐.avi

在图 15-2 所示的界面中，如果触摸按下♫图标则会获取 SD 卡中的音乐文件，在里面可以选择一个音乐文件作为闹钟铃声。此功能是通过文件 ChooseBellActivity.java 实现的，具体实现代码如下所示。

```
public class SetBellPreference extends ListPreference{
    private Uri mAlert;
    private PreferenceActivity preferenceActivity;
    private int mId;
    public SetBellPreference(Context context, AttributeSet attrs) {
        super(context, attrs);
        this.preferenceActivity=(PreferenceActivity)context;
        String[ ] values = context.getResources().getStringArray(R.array.choose_bell);
        mId=preferenceActivity.getIntent().getIntExtra(Alarms.ALARM_ID, -1);
        setEntries(values);
        setEntryValues(values);
    }
```

```
    @Override
    protected void onPrepareDialogBuilder(Builder builder) {
        CharSequence[ ] entries = getEntries();

        builder.setSingleChoiceItems(entries, -1, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                switch (which) {
                    case 0:
                        break;
                    case 1:
                        Intent intent = new Intent(preferenceActivity, ChooseBellActivity.class);
                        intent.putExtra(Alarms.ALARM_ID, mId);
                        intent.putExtra("TYPE", 1);
                        preferenceActivity.startActivity(intent);
                        preferenceActivity.finish();
                        break;
                    case 2:
                        Intent intent2 = new Intent(preferenceActivity, ChooseBellActivity.class);
                        intent2.putExtra(Alarms.ALARM_ID, mId);
                        intent2.putExtra("TYPE", 2);
                        preferenceActivity.startActivity(intent2);
                        preferenceActivity.finish();
                        break;
                    default:
                        break;
                }
            }
        });
    }

    @Override
    protected void onDialogClosed(boolean positiveResult) {
        if (positiveResult) {
            Alarm
        alarm=com.android.superdeskclock.Alarms.getAlarm(preferenceActivity. getContentResolver(),mId);
            alarm.silent=true;
            alarm.alert=Uri.parse("silent");
            mAlert=alarm.alert;
            ContentValues values = com.android.superdeskclock.Alarms.createContentValues(alarm);
            ContentResolver resolver = preferenceActivity.getContentResolver();
            resolver.update(ContentUris.withAppendedId(Alarm.Columns.CONTENT_URI, alarm.id),values,
null, null);

            setSummary(R.string.silent_alarm_summary);
        }
    }

    public void setAlert(Uri alert) {
        mAlert = alert;
        if (alert != null) {
            final Ringtone r = RingtoneManager.getRingtone(getContext(), alert);
```

```
                if (r != null) {
                    setSummary(r.getTitle(getContext()));
                }
            } else {
                setSummary(R.string.silent_alarm_summary);
            }
        }

        public Uri getAlert() {
            return mAlert;
        }
}
```

因为是在模拟器中运行，所以执行效果如图 15-8 所示。



图 15-8　执行效果

第 16 章　开发一个音乐播放器

随着硬件移动设备越来越先进，人们对移动设备的要求也越来越高，从追求技术到追求视觉，逐步提高了对系统的要求。Android 作为一个开源的系统，本章的音乐播放器实例就采用了 Android 开源系统技术，利用 Java 语言和 Eclipse 编辑工具对播放器进行编写，同时给出了详细的系统设计过程、部分界面图及主要功能运行流程图。在本章的内容中，还对开发过程中遇到的问题和解决方法进行了详细的讨论。本章讲解的音乐播放器实例集播放、暂停、停止、上一首、下一首、音量调节、歌词显示等功能于一体，性能良好，在 Android 系统中能独立运行。该播放器还拥有对手机文件浏览器的访问功能、歌曲播放模式以及歌词开闭状态的友好设置。“.MP3”的全名是 MPEG Audio Layer-3，是一种声音文件的压缩格式。因为本播放器只限于应用层程序的探讨，所以对具体的压缩算法不作深究。

16.1　项 目 介 绍

知识点讲解：光盘:视频\知识点\第 16 章\项目介绍.avi

本章播放器源码保存在本书附带光盘中的“光盘:\daima\16”目录下。在讲解具体编码之前，先简要介绍本项目的产生背景和项目目的知识，为后面的具体编码打好理论基础。

16.1.1　项目背景介绍

随着社会生活的节奏越来越快，人们对手机的要求也越来越高，由于手机市场发展迅速，使得手机操作系统也出现了不同的分类，现在的市场上主要有 3 个手机操作系统：Windows Mobile、Symbian，以及谷歌的 Android 操作系统，其中占有开放源代码优势的 Android 系统有最大的发展前景。那么能否在手机上拥有自己编写的个性音乐播放器呢？答案是完全可以！谷歌 Android 系统就能做到。本章讲解的音乐播放器实例就是基于谷歌 Android 手机平台的播放器。

随着计算机的广泛运用和手机市场的迅速发展，各种音频、视频资源也在网上广为流传，这些资源看似平常，但已经渐渐成为人们生活中必不可少的一部分。于是各种手机播放器也紧跟着发展起来，但是很多播放器一味追求外观花哨，功能庞大，对用户的手机造成了很多资源浪费，比如 CPU、内存等的占用率过高，在用户需要多任务操作时，受到了不小的影响，带来了许多不便，而对于大多数普通用户，许多功能用不上，形同虚设。针对以上各种弊端，本项目选择了开发多语种的音频视频播放器，将各种性能优化，继承播放器的常用功能，满足一般用户（如听歌、看电影）的需求，除了能播放常见格式的语音视频文件，高级功能还能播放 RMVB 格式的视频文件。此外，还能支持中文、英文等语言界面。

要研究各种市场上流行的手机播放器，了解它们各自的插件及编码方式，还有各种播放器播放的特别格式文件，分析各种编码的优缺点以及各种播放器本身存在的缺陷和特点，编写出功能实用、使

用方便快捷的播放器。目前已经实现的功能有能播放常见音频文件的功能，例如 MP3 和 WAV 等；拥有播放菜单，能选择播放清单，具备一般播放器的功能，如快进、快退、音量调节等。播放模式也比较完善，例如有单曲播放、顺序播放、循环播放和随机播放等模式。

16.1.2 项目的目的

当今社会人们的工作压力大，而欣赏音乐是最好的舒缓压力的方式之一。本项目的目的是开发一个可以播放主流音乐文件格式的播放器，该播放器的主要功能是播放 MP3、WAV 等多种格式的音乐文件，能够控制播放、暂停、停止、上一曲、下一曲音乐，也具备音量调节功能，并且具有很好的视觉外观，还具有播放列表和歌曲文件的管理操作等多种播放控制功能，界面简明，操作简单。

本项目是一款基于 Android 手机平台的音乐播放器，使 Android 手机拥有个性的多媒体播放器，显得更生动灵活，与人们更为接近，让手机主人随时随地处于音乐视频的旋律之中，使人们的生活更加多样化，也使设计者更加熟练 Android 的技术和其他在市场上的特点。

16.2 系统需求分析

知识点讲解：光盘:视频\知识点\第 16 章\系统需求分析.avi

根据项目的目标，我们可获得项目系统的基本需求，以下从不同角度来描述系统的需求，并且使用例图来描述，系统的功能需求分成四部分来概括，即播放器的基本控制需要、播放列表管理需求、播放器友好性需求和播放器扩展卡需求。

16.2.1 构成模块

本系统的构成模块如图 16-1 所示。

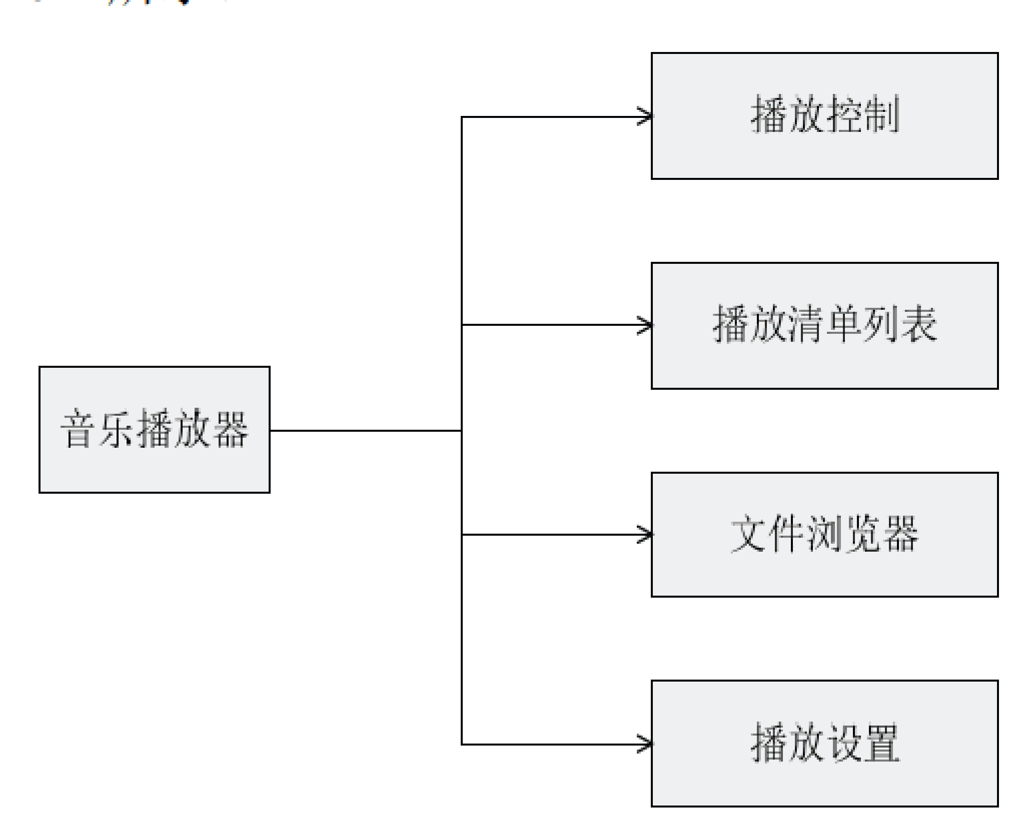
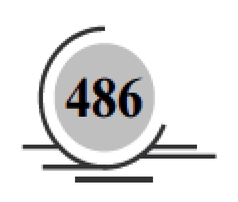


图 16-1 系统构成模块

各个模块的具体说明如下所示。

1. 播放控制模块

播放控制模块的功能是控制播放的音乐文件。



（1）播放

- 目标：使得用户可以播放在播放列表中选中的歌曲。
- 前置条件：播放器正在运行。
- 基本事件流：
  - 用户单击“播放”按钮。
  - 播放器将播放列表中的当前歌曲。

（2）暂停播放

- 目标：使得用户可以暂停正在播放的歌曲。
- 前置条件：歌曲正在播放且未停止和暂停。
- 基本事件流：
  - 单击“暂停”按钮。
  - 播放器将暂停当前的歌曲。

（3）停止播放

- 目标：使得用户可以停止正在播放的歌曲。
- 前置条件：歌曲正在播放或暂停。
- 基本事件流：
  - 用户单击“停止”按钮。
  - 播放器将停止当前播放的歌曲。

（4）上一首/下一首

- 目标：使得用户可以听上一首或下一首歌曲。
- 前置条件：歌曲正在播放或暂停。
- 基本事件流：
  - 用户单击“上一首”或“下一首”按钮。
  - 播放器将播放上一首或下一首歌曲。

（5）播放清单

- 目标：使得用户可以进入播放清单。
- 前置条件：程序在运行。
- 基本事件流：
  - 用户单击“清单”按钮。
  - 播放器进入清单列表。

播放控制的基本结构如图 16-2 所示。

2. 播放清单列表管理

当用户选中列表中的某一项歌曲后会显示播放清单，我们可以进行如下操作。

（1）播放

- 目标：使程序播放选中的歌曲。
- 前置条件：程序运行在播放菜单选项中。
- 基本事件流：
  - 用户单击“播放”按钮。
  - 播放器进入播放状态。

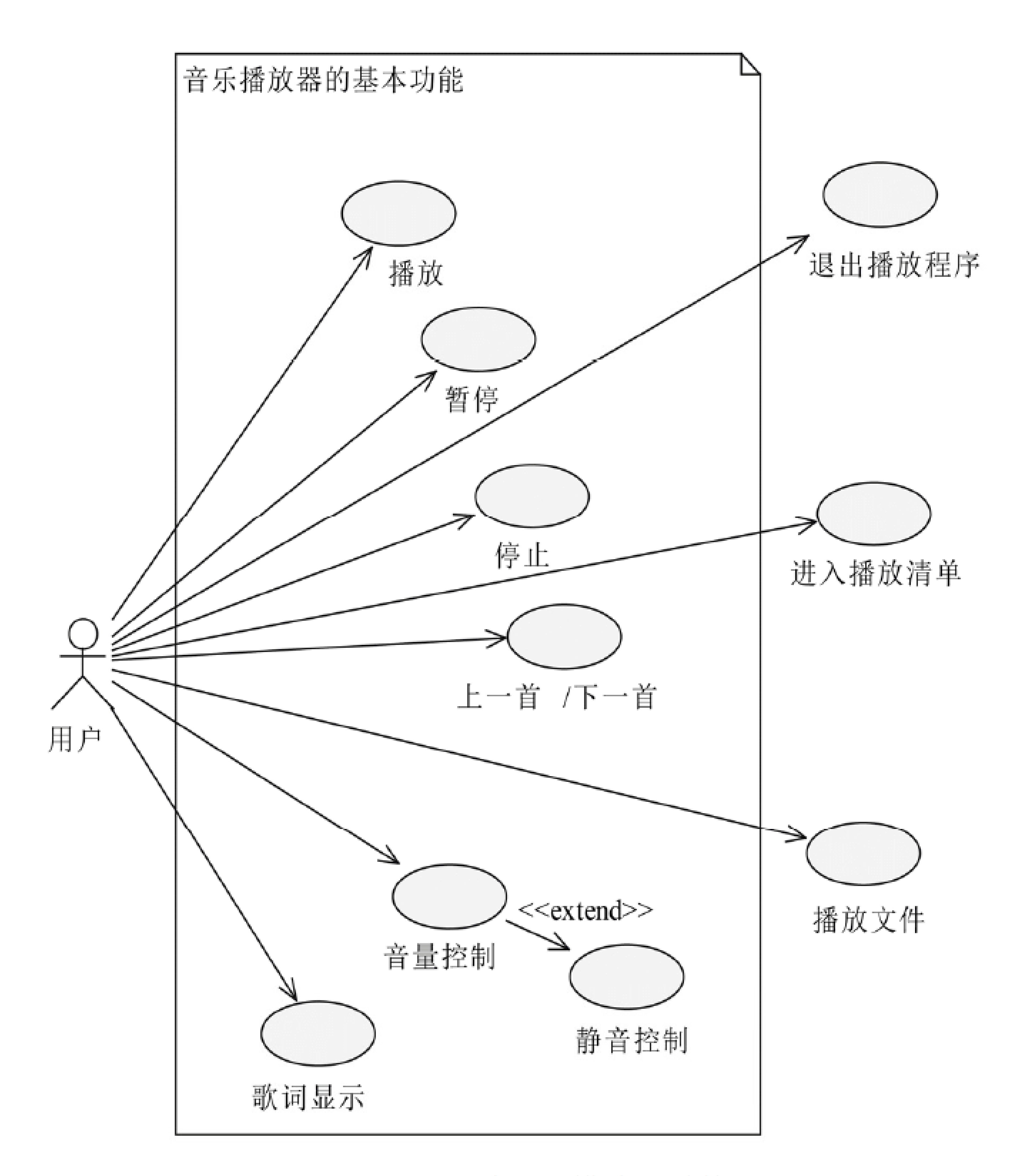


图 16-2　播放控制模块的结构

（2）视频详情

- 目标：使得程序显示歌曲详情。
- 前置条件：程序运行在播放菜单选项中。
- 基本事件流：
  - 用户单击“详细”按钮。
  - 显示歌曲详细状态。

（3）增加

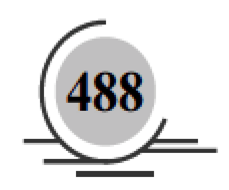
- 目标：使得程序进入手机扩展SD卡。
- 前置条件：程序运行在播放菜单选项中。
- 基本事件流：
  - 用户单击“增加”按钮。
  - 播放器进入手机扩展 SD 卡。

（4）移除/全部移除

- 目标：使选中的歌曲被移除。
- 前置条件：程序运行在播放菜单选项中。
- 基本事件流：
  - 用户单击“移除/全部移除”按钮。
  - 播放器移除选中歌曲/全部移除歌曲。

（5）设定

- 目标：使得程序进入播放器设定状态。



- ❑ 前置条件：程序运行在播放菜单选项中。
- ❑ 基本事件流：
  - ✧ 用户单击“设定”按钮。
  - ✧ 播放器进入设定界面。

本实例播放清单界面的结构如图 16-3 所示。

3. 播放设置模块

本模块用于设置音乐的播放方式，并设置是否显示歌词。

（1）播放模式

- ❑ 目标：使得程序进入播放模式设定状态。
- ❑ 前置条件：程序运行在播放器设定界面中。
- ❑ 基本事件流：
  - ✧ 用户单击“顺序”“随机”“单曲”按钮。
  - ✧ 播放器进入选中模式播放状态。

（2）显示歌词

- ❑ 目标：使得程序进入播放器歌词设置状态。
- ❑ 前置条件：程序运行在播放器设定界面中。
- ❑ 基本事件流：
  - ✧ 用户单击“歌词开关按钮”按钮。
  - ✧ 播放器显示或关闭歌词。

本实例设置的结构如图 16-4 所示。

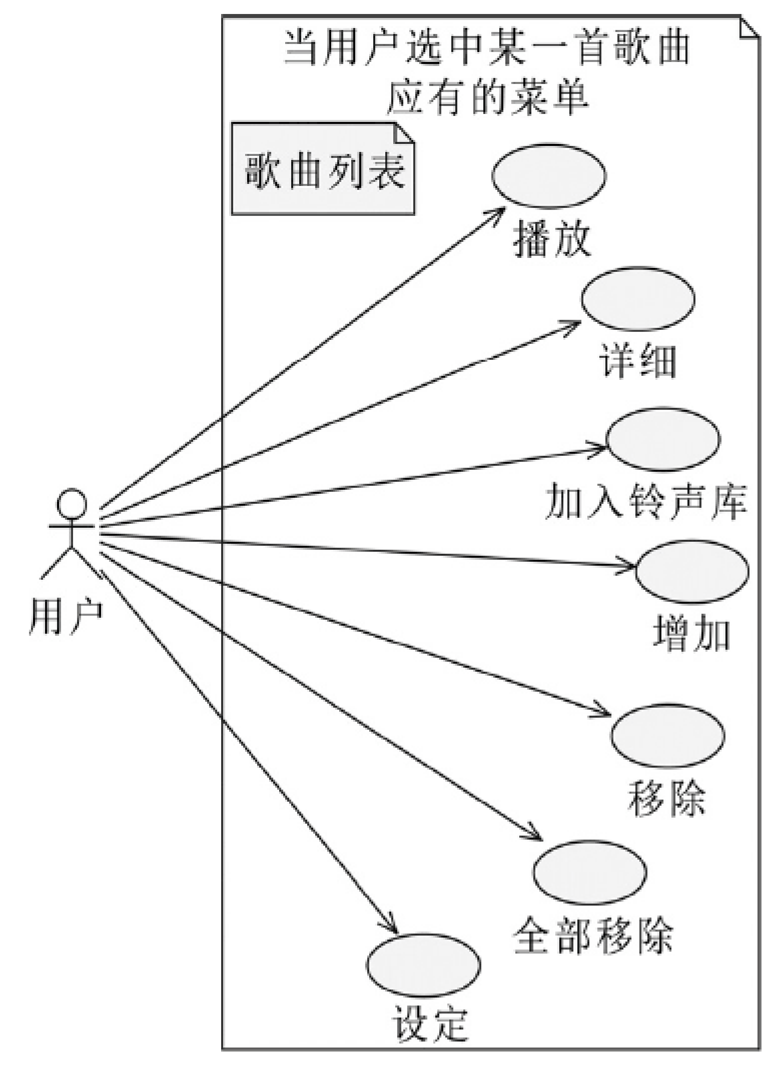


图 16-3　播放清单界面结构

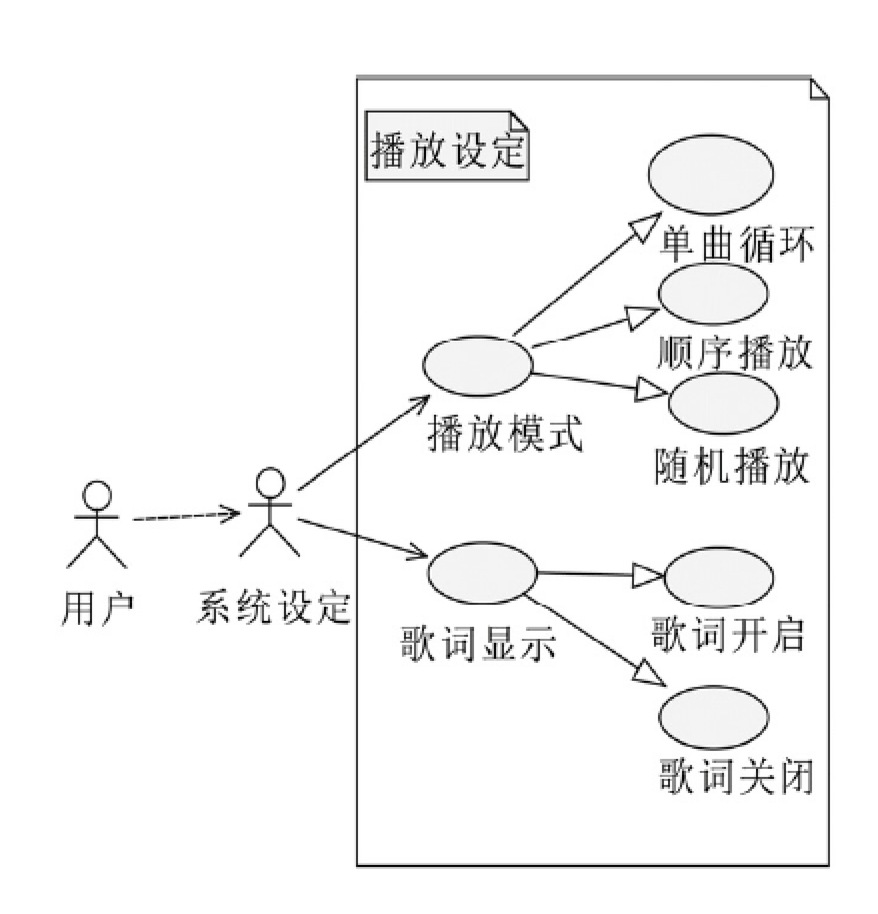


图 16-4　设置界面结构

4. 文件浏览器

此模块的功能是浏览系统内或 SD 卡中的文件信息。

（1）SDcard

- ❑ 目标：使得程序进入SDcard目录。
- ❑ 前置条件：程序运行在目录界面中。

❑ 基本事件流：
  ✧ 用户单击 SDcard 选项。
  ✧ 程序进入 SDcard 目录。

（2）System

❑ 目标：使得程序进入System目录。
❑ 前置条件：程序运行在目录界面中。
❑ 基本事件流：
  ✧ 用户单击 System 选项。
  ✧ 程序进入 System 目录下。

本实例文件浏览器模块的结构如图 16-5 所示。

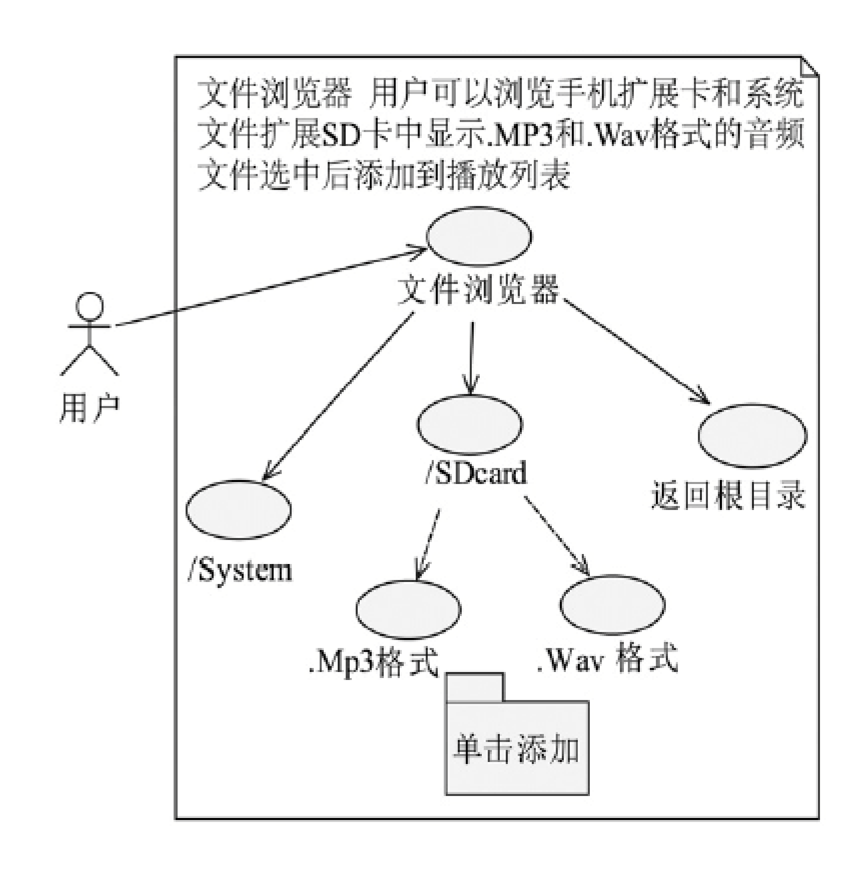


图 16-5 文件浏览器模块结构

16.2.2 系统流程

本章音乐播放器的系统流程如图 16-6 所示。

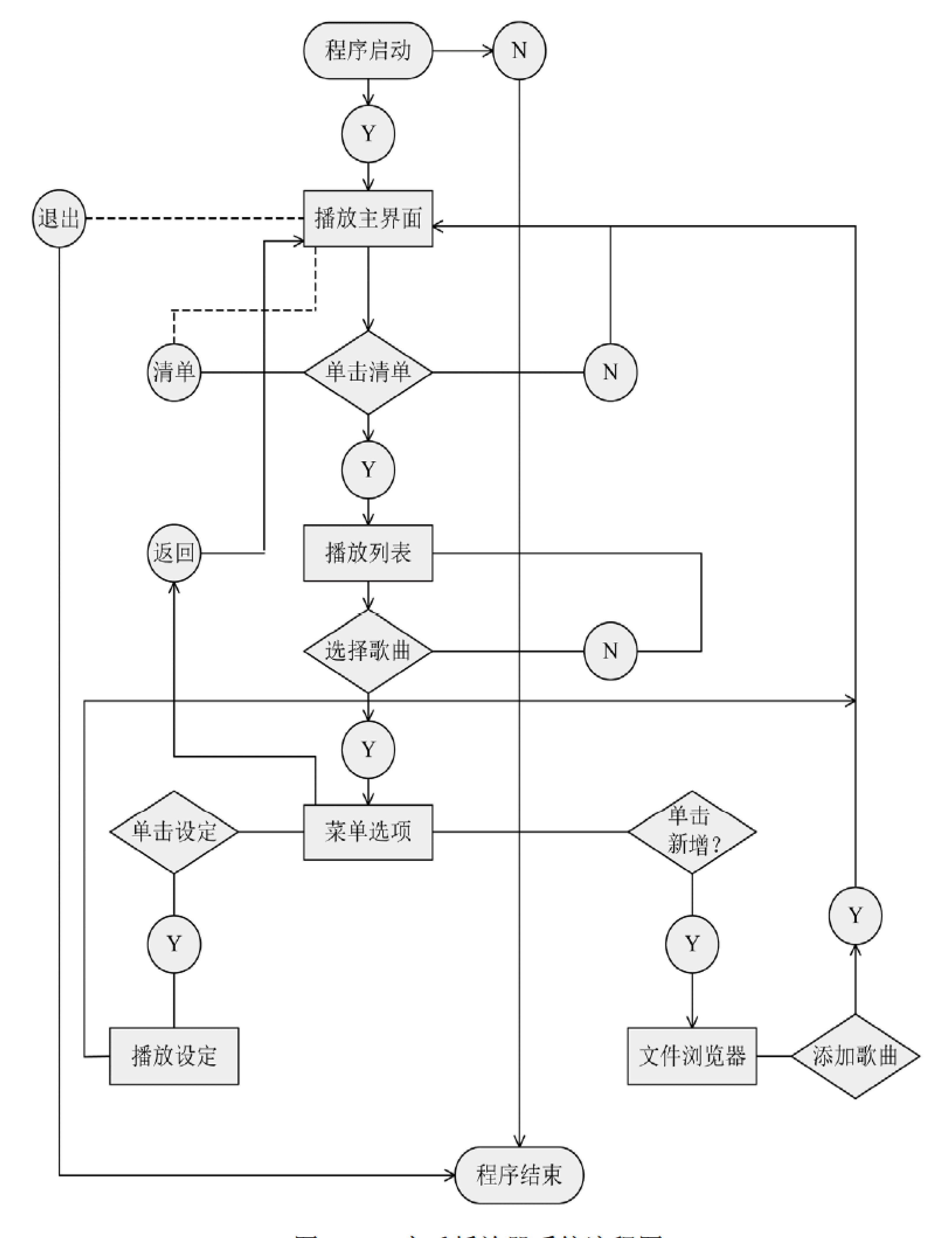


图 16-6 音乐播放器系统流程图

16.2.3　功能结构图

本章音乐播放器系统的完整功能结构如图 16-7 所示。

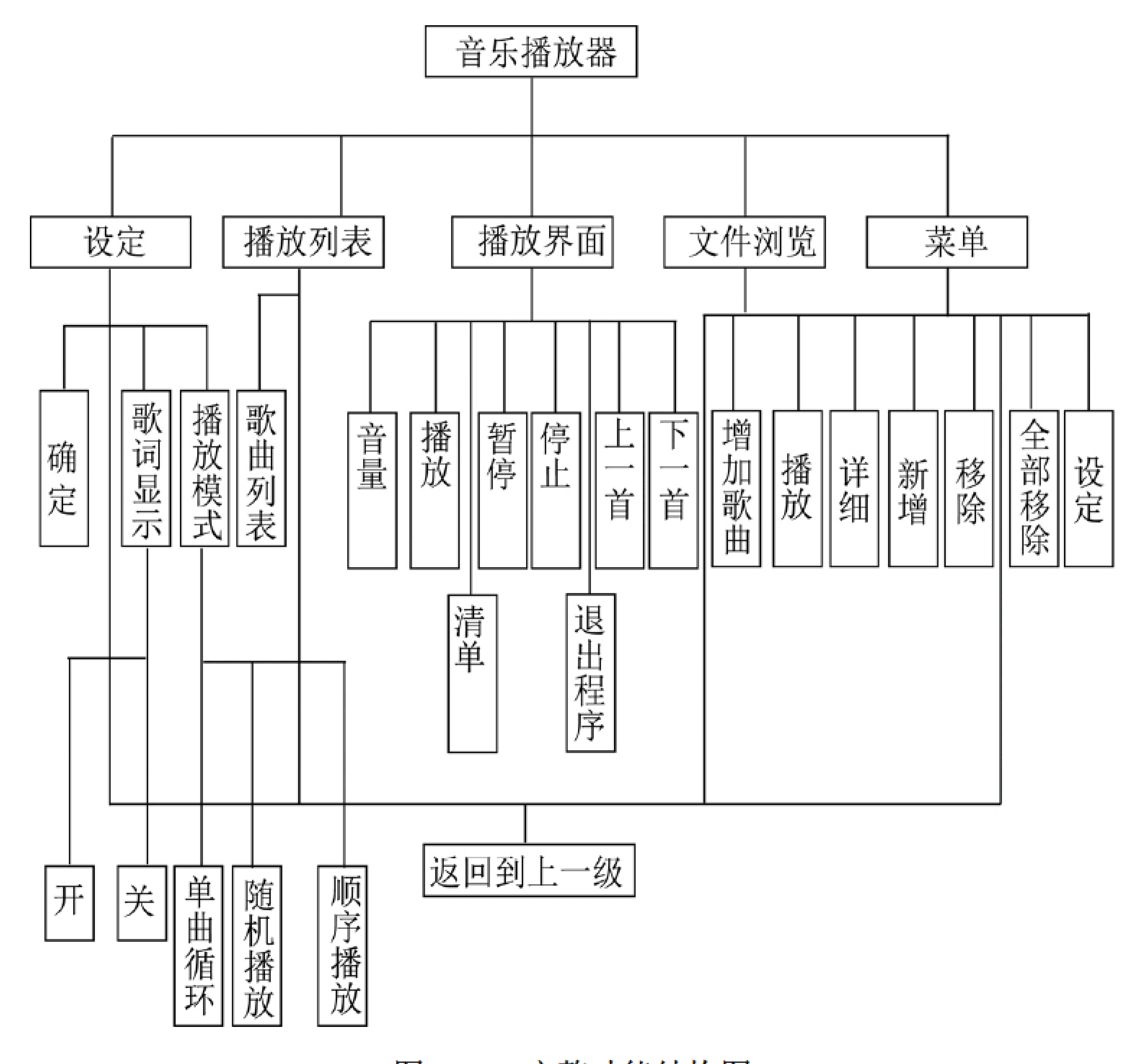


图 16-7　完整功能结构图

16.2.4　系统功能说明

本章音乐播放器系统各个模块功能的说明如表 16-1 所示。

表 16-1　模块结构功能说明信息表

功能类别	子功能	子功能
播放列表	播放列表菜单	退出播放
		从扩展卡寻找歌曲
	歌曲菜单	播放→进入播放界面
		删除→数据库同步更新
		重命名→数据库同步更新
		向上、下移动→数据库同步更新
播放界面	播放	播放歌曲→线程启动→时间更新
	暂停	暂停歌曲→线程暂停→时间暂停
	停止	停止歌曲→线程停止→时间停止
	上一首	播放列表索引变化→寻找上一 ID 歌曲
	下一首	播放列表索引变化→寻找下一 ID 歌曲

续表

功能类别	子 功 能	子 功 能
播放界面	播放界面菜单	返回到播放列表
		返回到主菜单
		从扩展卡寻找歌曲
		退出播放器
		隐藏播放界面
主菜单	退出程序	程序退出
	进入播放列表	显示播放列表

16.2.5 系统需求

（1）系统界面需求

播放器界面要求布局合理，颜色舒适，控制按钮友好，为了减少开发工程量，图片素材多数为公司项目素材，如图 16-8 所示。

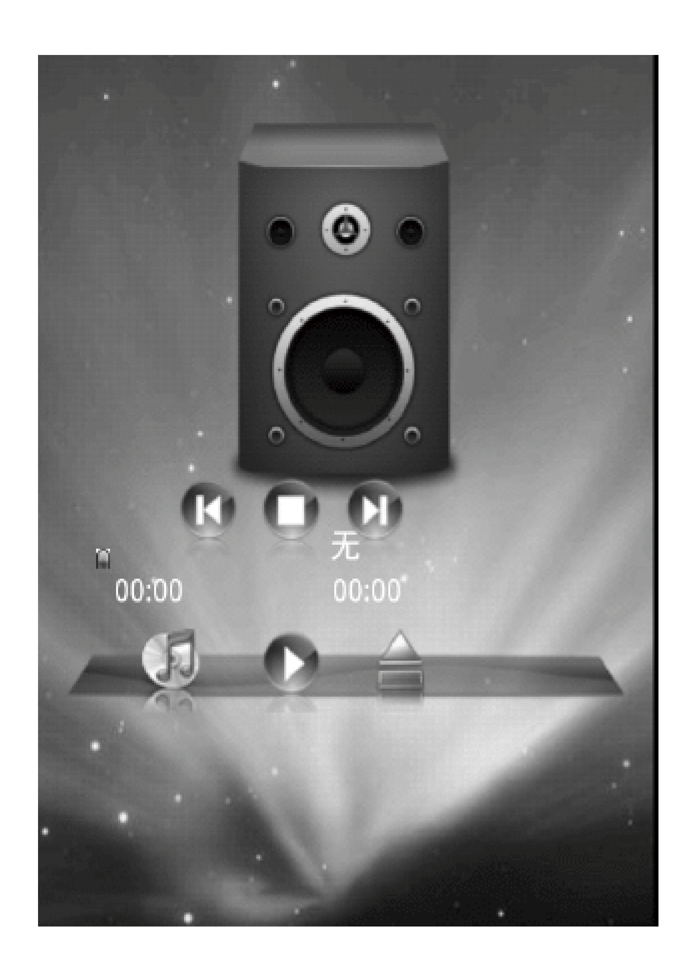


图 16-8　播放器主界面

（2）系统性能需求

根据 Android 手机系统要求无响应时间为 5 秒，所以就有如下性能要求：

- 当要求歌曲播放时，程序响应时间最长不能超过5秒。
- 当要求歌曲暂停时，程序响应时间最长不能超过5秒。
- 当要求歌曲停止时，程序响应时间最长不能超过5秒。
- 当要求歌曲上/下一首时，程序响应时间最长不能超过5秒。
- 当要求进行清单列表时，程序响应时间最长不能超过5秒。

（3）运行环境需求

- 操作系统：Android手机基于Linux操作系统。
- 支持环境：Android 1.5 - 2.0.1版本。
- 开发环境：Eclipse 3.5 ADT 0.95。

16.3 数据库设计

知识点讲解：光盘:视频\知识点\第 16 章\数据库设计.avi

数据库是存放数据的仓库，只不过这个仓库是在计算机存储设备上，而且数据是按一定的格式存放的。数据库中的数据按一定数据模型组织、描述和存储，具有较小的重复度、较高的数据独立性和易扩展性，并且可以被在一定范围内的各种用户共享。在涉及数据库的软件开发中，需要根据有待解决的问题性质、规模，以及所采用的前端程序创建工具等，作出合适的数据库类型选择。

16.3.1 字段设计

字段 file_table 用于保存歌曲的名字、类型和路径，具体说明如表 16-2 所示。



表 16-2　字段 file_table 说明

属　　性	数 据 类 型	说　　明
_Id	INTEGER	id 号
fileName	TEXT	歌曲名字
filePath	TEXT	歌曲路径
sort	INTEGER	歌曲类型

SD 卡中保存歌曲用表 16-3 中的字段来保存。

表 16-3　歌曲详情表

属　　性	数 据 类 型	说　　明
_ID	INTEGER	id 号
TITLE	TEXT	标题
ARTIST	TEXT	艺术家
ALBUM	TEXT	专辑
SIZE	LONG	大小

在 Android 系统中，通过自带的 MediaStore 封闭类来存储媒体信息，通过 Uri EXTERNAL_CONTENT_URI 来访问 SDcard 中的歌曲详细信息。

16.3.2　E-R 图设计

音乐播放器的 E-R 图如图 16-9 所示。

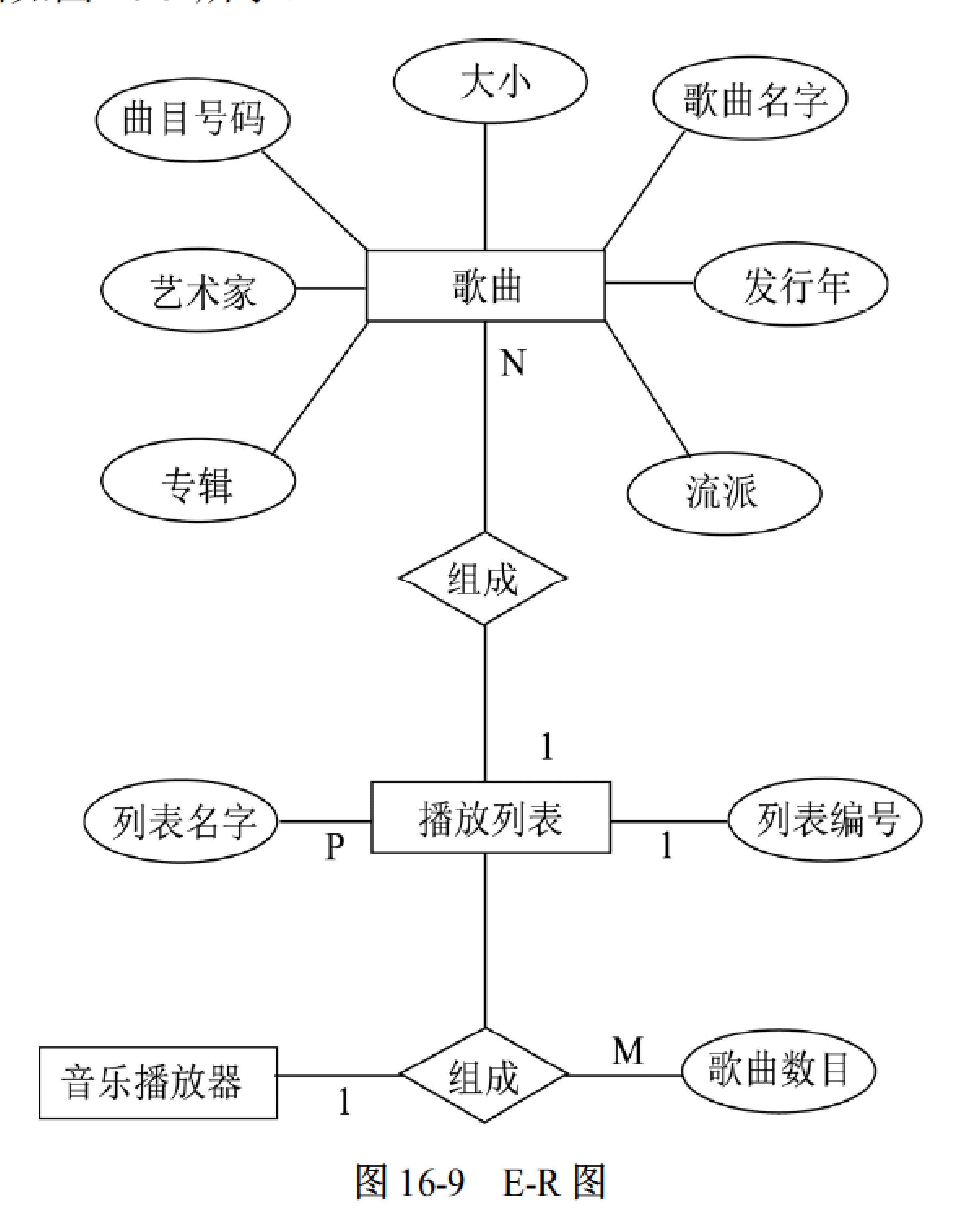


图 16-9　E-R 图

16.3.3 数据库连接

在 Android 系统中自带 iSQLite 数据库，这是一个十分小型的数据库，这样正适合 Android 这种移动平台使用。Android 数据库存储的位置在“data/data/<项目文件夹>/databases/”目录下。

Android 使用 ContentProvider 作为内容提供商，SQLiteOpenHelper 数据库帮助类来进行对数据库的创建和操作。通过 Context.getContentResolver()方法直接对数据库进行操作。程序中数据库类为 DBHelper extends SQLiteOpenHelper（继承关系），内容提供类为 DBProvider extends ContentProvider（继承关系）。

16.3.4 创建数据库

Android 提供了标准的数据库创建方式。继承自 SQLiteOpenHelper，实现 onCreate 和 onUpgrade 两个方法，这样的好处是便于数据库版本的升级。在此编写文件 DBHelper.java，实现连接数据库的算法，具体代码如下所示。

```
public class DBHelper extends SQLiteOpenHelper{
      /**
       * 数据库名称常量
       */
      private static final String DATABASE_NAME = "MyMusic.db";
      /**
       *   数据库版本常量
       */
      private static final int DATABASE_VERSION = 1;
      /**
       *   表名称常量
       */
      public static final String TABLES_TABLE_NAME = "File_Table";
      private static final String DATABASE_CREATE = "CREATE TABLE " + FileColumn.TABLE +" ("
       + FileColumn.ID+" integer primary key autoincrement,"
       + FileColumn.NAME+" text,"
       + FileColumn.PATH+" text,"
       + FileColumn.SORT+" integer,"
       + FileColumn.TYPE+" text)";
       /**
        *   构造方法
        * @param context
        */
       public DBHelper(Context context) {
              //创建数据库
              super(context, DATABASE_NAME,null, DATABASE_VERSION);
       }
       /**
        *   创建时调用
        */
       public void onCreate(SQLiteDatabase db) {
              /*Locale l = new Locale("zh", "CN");
```

```
            db.setLocale(l);*/
        db.execSQL(DATABASE_CREATE);
    }
    /**
     *   版本更新时调用
     */
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            //删除表
            db.execSQL("DROP TABLE IF EXISTS File_Table");
        onCreate(db);
    }
}
```

如果创建数据库不成功则抛出 FIleNotFoundException 异常。

16.3.5　操作数据库

Android 对数据库的操作主要有插入、删除、更新、查询操作，在进行任何操作时都必须指定一个 Uri，才能对相应的表进行数据操作。编写文件 DBProvider.java，在里面分别编写数据插入、修改、查询和删除操作的实现方法，具体代码如下所示。

```
public class DBProvider extends ContentProvider {
    private DBHelper dbOpenHelper;
    public static final String AUTHORITY = "MUSIC";
    public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY
                + "/" + FileColumn.TABLE);
    @Override
    public int delete(Uri arg0, String arg1, String[ ] arg2) {
        SQLiteDatabase db = dbOpenHelper.getWritableDatabase();

        try {
            db.delete(FileColumn.TABLE, arg1, arg2);
            Log.i("info", "delete");
        } catch (Exception ex) {
            ex.printStackTrace();
            Log.e("error", "delete");
        }
        return 1;
    }
    /**
     * 待实现
     */
    @Override
    public String getType(Uri uri) {
        return null;
    }
    /**
     * 插入
     */
    @Override
    public Uri insert(Uri uri, ContentValues values) {
```

```
            SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
            long count = 0;
            try {
                count = db.insert(FileColumn.TABLE, null, values);
            } catch (Exception ex) {
                ex.printStackTrace();
                Log.e("error", "insert");
            }
            if (count > 0)
                return uri;
            else
                return null;
        }
        @Override
        public boolean onCreate() {
            dbOpenHelper = new DBHelper(getContext());
            return true;
        }
        /**
         * 根据条件查询
         * @return 数据集
         */
        @Override
        public int update(Uri uri, ContentValues values, String selection,
                String[ ] selectionArgs) {
            SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
            int i = 0;
            try {
                i = db.update(FileColumn.TABLE, values, selection, null);
                return i;
            } catch (Exception ex) {
                Log.e("error", "update");
            }
            return 0;
        }
}
```

16.3.6 数据显示

本项目在显示数据时，利用 Cursor 游标类指向数据表中的某一项，然后进行查询数据，并用 Log 日志显示出来。

```
//数据库查询操作
@Override
    public Cursor query(Uri uri, String[ ] projection, String selection,String[ ] selectionArgs, String sortOrder) {
        SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
//依次参数为：表名，查询字段，where 语句，替换
//group by(分组)，having(分组条件),order by(排序)
        Cursor cur = db.query(FileColumn.TABLE, projection, selection,selectionArgs, null, null, sortOrder);
return cur;}
```

16.4　具 体 编 码

知识点讲解：光盘:视频\知识点\第 16 章\具体编码.avi

经过前面内容的讲解，本播放器实例项目的前期工作已经结束。在接下来的内容中，将详细讲解本项目的具体编码过程。

16.4.1　设置服务信息

编写文件 SystemService.java，在此设置项目的服务信息，主要代码如下所示。

```
public class SystemService {
	private Context context;
	private Cursor cursor;
	public SystemService(Context context) {
		this.context = context;
	}

	public Cursor allSongs() {
		if (cursor != null)
			return cursor;
		ContentResolver resolver = context.getContentResolver();
		cursor = resolver.query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
				null, null, null, MediaStore.Audio.Media.DEFAULT_SORT_ORDER);
		return cursor;
	}

	/**
	 * 读取正在播放歌曲的艺术家
	 * @return
	 */
	public String getArtist() {
		return cursor.getString(cursor
				.getColumnIndexOrThrow(MediaStore.Audio.Media.ARTIST));
	}
	/**
	 * 读取正在播放的歌曲名字
	 * @return 歌曲名字
	 */
	public String getTitle() {
		String title = cursor.getString(cursor
				.getColumnIndexOrThrow(MediaStore.Audio.Media.TITLE));
		try {
			title=EncodingUtils.getString(title.getBytes(), "UTF-8");
		} catch (Exception e) {

			e.printStackTrace();
		}
```

```
            return title;
        }

        /**
         *  读取正在播放歌曲的专辑
         * @return 专辑名
         * @throws RemoteException
         */
        public String getAlbum() throws RemoteException {
            return cursor.getString(cursor
                    .getColumnIndexOrThrow(MediaStore.Audio.Media.ALBUM));
        }

        /*public int getDuration() throws RemoteException {
            //获得当前歌曲的时长
            return player.getDuration();
        }public int getTime() throws RemoteException {
            //获得当前的媒体时间
            return player.getCurrentPosition();
        }
```

16.4.2 播放器主界面

Android 的每一个可视化界面，都有其唯一的布局配置文件，该文件里面有各种布局方式和各种资源文件，如图像、文字、颜色的引用，程序在运行时，可以通过代码对各配置文件进行读取。这样就可以形成不同的可视化界面和炫丽的效果。

（1）本实例主界面的布局文件是 main.xml，主要代码如下所示。

```
<TextView
    android:id="@+id/current_music"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"
    android:textColor="#ffffff"
     android:padding="10dip"
     android:cacheColorHint="#00000000"
    android:text="this is TextView…"
    android:layout_y="320px"/>
  -->
<Gallery android:id="@+id/gallery"
        android:layout_width="fill_parent"
                android:layout_height="200dp"
                android:layout_alignParentLeft="true"
                android:spacing="16dp"
                android:cacheColorHint="#00000000"
                android:layout_centerVertical="true"
                android:layout_y="20px"/>
<SeekBar android:id="@+id/seekbar" android:layout_width="245px"
            android:layout_height="20px" android:layout_x="40px"
        android:progressDrawable="@drawable/seekbar_style"
```

```
        android:thumb="@drawable/thumb"
            android:paddingLeft="18px"
            android:paddingRight="15px"
            android:paddingTop="5px"
            android:paddingBottom="5px"
            android:progress="0"
android:max="100"
android:secondaryProgress="0"
android:layout_y="350px"/>
<TextView android:layout_x="60px" android:layout_height="wrap_content"
            android:text="00:00" android:layout_y="370px" android:id="@+id/current_time_text"
            android:layout_width="wrap_content"></TextView>
      <TextView android:layout_x="230px" android:layout_height="wrap_content"
            android:text="00:00" android:layout_y="370px" android:id="@+id/end_Time_Text"
            android:layout_width="wrap_content"></TextView>
<LinearLayout android:orientation="horizontal"
             android:gravity="center"
             android:layout_y="423px"
             android:layout_height="wrap_content"
              android:layout_width="fill_parent"
              android:background="@drawable/buttonground" />
<!-- 建立第一个 ImageButton -->
  <ImageButton
  android:id="@+id/btStart"
   android:layout_height="70dp"
  android:layout_width="70dp"
  android:layout_x="145px"
  android:layout_y="390px"
  android:background="#00000000"
  android:src="@drawable/play"
  />
  <!-- 建立第二个 ImageButton
  <ImageButton
  android:id="@+id/pause"
  android:layout_height="wrap_content"
  android:layout_width="wrap_content"
 android:background="#00000000"
  android:layout_x="141px"
  android:layout_y="50px"
  android:src="@drawable/pause"
  />
  <!-- 建立第三个 ImageButton -->
  <ImageButton
  android:id="@+id/before"
  android:layout_height="70dp"
  android:layout_width="70dp"
  android:background="#00000000"
  android:layout_x="80px"
  android:layout_y="280px"
  android:src="@drawable/backward"
  />
```

```
  <!-- 建立第四个 ImageButton -->
  <ImageButton
  android:id="@+id/next"
  android:layout_height="70dp"
  android:layout_width="70dp"
  android:background="#00000000"
  android:layout_x="210px"
  android:layout_y="280px"
  android:src="@drawable/forward"
  />
  <!-- 建立第五个 ImageButton -->
  <ImageButton
  android:id="@+id/btStop"
  android:layout_height="70dp"
  android:layout_width="70dp"
  android:layout_x="145px"
  android:layout_y="280px"
     android:background="#00000000"
  android:src="@drawable/stop"
  />

<ImageButton
  android:id="@+id/listplay"
  android:layout_height="70dp"
  android:layout_width="70dp"
 android:cacheColorHint="#00000000"
  android:layout_x="50px"
  android:layout_y="390px"
   android:background="#00000000"
  android:src="@drawable/itunes2" />
  <!--
  <ImageButton
  android:id="@+id/player"
  android:layout_height="70dp"
  android:layout_width="70dp"
 android:cacheColorHint="#00000000"
  android:layout_x="140px"
  android:layout_y="390px"
   android:background="#00000000"
  android:src="@drawable/wmp2" />
   -->
  <ImageButton
  android:id="@+id/returnBt"
  android:layout_height="70dp"
  android:layout_width="70dp"
 android:cacheColorHint="#00000000"
  android:layout_x="230px"
  android:layout_y="390px"
   android:background="#00000000"
  android:src="@drawable/white"
/>
```



（2）播放器主界面是一个 Activity，Android 工程在每个 activity 启动的时候会首先执行 Oncreate()方法。本实例主界面的程序文件是 MainPlayActivity.java，其具体实现流程如下所示。

- 实现界面初始化工作，如果有播放的歌曲则在播放器中显示歌曲名，并显示上次的播放进度。如果设置了显示歌词，则还会在界面中显示歌词。对应代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		requestWindowFeature(Window.FEATURE_NO_TITLE);
		setContentView(R.layout.main);
		systemProvider=new SystemService(this);
		cursor=systemProvider.allSongs();
		SharedPreferences sp = getSharedPreferences("MUSIC",MODE_WORLD_READABLE);
		if (sp != null) {
			playingName = sp.getString("PLAYINGNAME", null);
			selectName = sp.getString("SELECTNAME", null);
			String s = sp.getString("MUSIC_LIST", null);
			if (s != null)
				music_List = StringHelper.spiltString(s);
		}

		init_Play_Rack();                                         //界面初始化
		if (playingName != null) {//
			int time1 = mplayer.getDuration();
			int time2 = mplayer.getCurrentPosition();
			seekBar.setMax(time1);
			seekBar.setProgress(time2);
			currently_Time.setText(getFileTime(time2));
			end_Time.setText(getFileTime(time1));
			currently_Music.setText(playingName);


			handler.removeCallbacks(thread_One);
			handler.postDelayed(thread_One, 1000);
			lrc_time = new ArrayList<String>();
			lrc_word = new ArrayList<String>();
			showLrc(playingName);                                 //歌词显示
		}
		if (selectName != null) {                                 //播放选中的歌曲
			play_bt.setImageBitmap(musicAdapter.getSuspend_Icon()); //默认暂停图标
			play_Music();
			lrc_time = new ArrayList<String>();
			lrc_word = new ArrayList<String>();
			showLrc(selectName);                                  //歌词显示
		}
		if (!(currently_Music.getText().toString()).equals("无")) {
			play_bt.setOnTouchListener(playListener);             //播放监听器
			seekBar.setOnSeekBarChangeListener(seekBarListener);  //音轨监听器
			stop_bt.setOnTouchListener(stopListener);             //停止监听器
			move_Down.setOnTouchListener(downListener);           //下一首歌曲监听器
			move_Up.setOnTouchListener(upListener);               //上一首歌曲监听器
		}
```

```
        list_bt.setOnTouchListener(list_bt_listener);                //清单监听器
        back_bt.setOnTouchListener(return_bt_listener);
        mplayer.setOnCompletionListener(playerListener);          //监听歌曲是否播放完

        mSwitcher = (ImageSwitcher) findViewById(R.id.switcher);
      mSwitcher.setFactory(this);
      mSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_in));
      mSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_out));
      mSwitcher.setImageResource(R.drawable.background);
      Gallery g = (Gallery) findViewById(R.id.gallery);
      g.setAdapter(new ImageAdapter(this));
      g.setSelection(200);
      g.setOnItemSelectedListener(this);
    }
```

- 设置暂停和重置处理，对应代码如下所示。

```
protected void onPause() {
        super.onPause();
        SharedPreferences sp = getSharedPreferences("MUSIC",
                MODE_WORLD_WRITEABLE);
        SharedPreferences.Editor editor = sp.edit();
        playingName = currently_Music.getText().toString();
        if (!playingName.equals("无"))
            editor.putString("PLAYINGNAME", playingName);
        editor.putString("SELECTNAME", selectName);
        editor.putString("MUSIC_LIST", StringHelper.toStringAll(music_List));
        editor.commit();
        handler.removeCallbacks(thread_One);
    }
    @Override
    protected void onResume() {
        super.onResume();
        systemProvider=new SystemService(this);
        cursor=systemProvider.allSongs();
        SharedPreferences sp = getSharedPreferences("MUSIC",MODE_WORLD_READABLE);
        if (sp != null) {
            playingName = sp.getString("PLAYINGNAME", null);
            selectName = sp.getString("SELECTNAME", null);
            String s = sp.getString("MUSIC_LIST", null);
            if (s != null)
                music_List = StringHelper.spiltString(s);
        }
        if (mplayer.isPlaying()) {
            handler.removeCallbacks(thread_One);
            handler.postDelayed(thread_One, 1000);
        }
        else
            handler.removeCallbacks(thread_One);
    }
```

```
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.i("info", "onDestroy");
        new File("/data/data/com.Rain.music.activity/shared_prefs/MUSIC.xml")
                .delete();
        new File("/data/data/com.Rain.music.activity/shared_prefs/SET_MSG.xml")
                .delete();
        System.exit(0);
    };
```

- 定义方法init_Play_Rack()，实现主界面的初始化处理，对应代码如下所示。

```
    private void init_Play_Rack() {
        list_bt = (ImageButton) findViewById(R.id.listplay);
        back_bt = (ImageButton) findViewById(R.id.returnBt);
        stop_bt = (ImageButton) findViewById(R.id.btStop);
        play_bt = (ImageButton) findViewById(R.id.btStart);
        move_Up = (ImageButton) findViewById(R.id.before);
        move_Down = (ImageButton) findViewById(R.id.next);
        end_Time = (TextView) findViewById(R.id.end_Time_Text);
        //title_Music = (TextView) findViewById(R.id.title_music);
        currently_Time = (TextView) findViewById(R.id.current_time_text);
        currently_Music = (TextView) findViewById(R.id.current_music);
        seekBar = (SeekBar) findViewById(R.id.seekbar);

        mplayer = MusicHelp.getMediaPlayer();
        musicAdapter = new MusicAdapter(this, music_List);
        handler = MusicHelp.getHandler();
        currently_Music.setText("无");
        currently_Music.setTextColor(Color.WHITE);
        currently_Time.setTextColor(Color.WHITE);
        end_Time.setTextColor(Color.WHITE);
        lrcTime = (TextView) findViewById(R.id.lrcText);
        SharedPreferences sp = getSharedPreferences("SET_MSG",
                MODE_WORLD_READABLE);
        if (sp != null) {
            if (sp.getString("sigle_Play", null) != null) {
                play_Mode = sp.getString("sigle_Play", null);
            }
            if (sp.getString("order_Play", null) != null) {
                play_Mode = sp.getString("order_Play", null);
            }
            if (sp.getString("random_Play", null) != null) {
                play_Mode = sp.getString("random_Play", null);
            }
            if (sp.getString("lyLrc", null) != null) {
                lrc_Show = sp.getString("lyLrc", null);
            }
            Log.i("info", "play_Mode=" + play_Mode);
            Log.i("info", "lrc_Show=" + lrc_Show);
        }
    }
```

❑ 定义方法onCompletion()，实现歌曲播放完监听器功能，对应代码如下所示。

```
OnCompletionListener playerListener = new OnCompletionListener() {
	@Override
	public void onCompletion(MediaPlayer mp) {

		play_Mode();// 播放模式
		lrc_time = new ArrayList<String>();
		lrc_word = new ArrayList<String>();
		showLrc(selectName);
	}
};
```

❑ 定义return_bt_listener，实现结束监听器处理，对应代码如下所示。

```
OnTouchListener return_bt_listener = new OnTouchListener() {
	@Override
	public boolean onTouch(View v, MotionEvent event) {
		if (event.getAction() == MotionEvent.ACTION_DOWN) {
			back_bt.setImageResource(R.drawable.whitepress);
		} else if (event.getAction() == MotionEvent.ACTION_UP) {
			back_bt.setImageResource(R.drawable.white);
			finish();
			onDestroy();
//	android.os.Process.killProcess(android.os.Process.myPid());
//获取 PID，目前获取自己的也只有该 API，否则从自己的/proc 中枚举其他进程
		//	System.exit(0);
		}
		return false;
	}
};
```

❑ 定义list_bt_listener，实现清单监听器的处理，对应代码如下所示。

```
OnTouchListener list_bt_listener = new OnTouchListener() {
	@Override
	public boolean onTouch(View v, MotionEvent event) {
		if (event.getAction() == MotionEvent.ACTION_DOWN) {
		//	v.setBackgroundResource(R.drawable.share_pressed);
			list_bt.setImageResource(R.drawable.itunespress);
		} else if (event.getAction() == MotionEvent.ACTION_UP) {
			//v.setBackgroundResource(R.drawable.share);
			list_bt.setImageResource(R.drawable.itunes2);

			Intent intent = new Intent(MainPlayActivity.this,
					PlayListActivity.class);
			startActivityForResult(intent, 0);
		}
		return false;
	}
}
```

❑ 定义OnSeekBarChangeListener seekBarListener，实现音轨监听器处理操作，对应代码如下所示。

```
private OnSeekBarChangeListener seekBarListener = new OnSeekBarChangeListener() {
	@Override
```

```
        public void onProgressChanged(SeekBar seekBar, int progress,
                boolean fromUser) {
            if (fromUser) {
                mplayer.seekTo(progress);
                currently_Time.setText(getFileTime(progress));
            }
        }
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        }
};
```

- 定义playListener，实现播放监听器的操作处理，对应代码如下所示。

```
OnTouchListener playListener = new OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    if (mplayer.isPlaying()) {
                        play_bt.setImageResource(R.drawable.pausepress);
                    }
                    else {
                        play_bt.setImageResource(R.drawable.playpress);
                    }
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    if (mplayer.isPlaying()) {
                        mplayer.pause();//暂停
                        currently_Time.setText(currently_Time.getText().toString());
                        play_bt.setImageBitmap(musicAdapter.getPlay_Icon());
                        handler.removeCallbacks(thread_One);
                    }
                    else {
                        if (is_stopping) {
                            play_Music();
                            is_stopping = false;
                            play_bt.setImageBitmap(musicAdapter.getSuspend_Icon());
                        } else {
                            mplayer.start();
                            handler.postDelayed(thread_One, 1000);
                            play_bt.setImageBitmap(musicAdapter.getSuspend_Icon());
                        }
                    }
                }
                return false;
            }
};
```

- 定义stopListener，实现停止监听器的处理，对应代码如下所示。

```
OnTouchListener stopListener = new OnTouchListener() {
            @Override
```

```
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    stop_bt.setImageResource(R.drawable.stoppress);
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    stop_bt.setImageResource(R.drawable.stop);
                    mplayer.stop();
                    currently_Time.setText("00:00");
                    end_Time.setText("00:00");
                    play_bt.setImageBitmap(musicAdapter.getPlay_Icon());
                    handler.removeCallbacks(thread_One);
                    seekBar.setProgress(1);
                    is_stopping = true;
                    lrcTime.setText("无");
                }
                return false;
            }
        };
```

❑ 定义downListener，实现下一首歌曲监听器的操作处理，对应代码如下所示。

```
OnTouchListener downListener = new OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                move_Down.setImageResource(R.drawable.forwardpress);

            } else if (event.getAction() == MotionEvent.ACTION_UP) {
                move_Down.setImageResource(R.drawable.forward);
                move_Down(currently_Music.getText().toString());
                lrc_time = new ArrayList<String>();
                lrc_word = new ArrayList<String>();
                showLrc(currently_Music.getText().toString());//歌词显示
            }
            return false;
        }
};
```

❑ 定义upListener，实现上一首歌曲监听器的操作处理，对应代码如下所示。

```
OnTouchListener upListener = new OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                move_Up.setImageResource(R.drawable.backwardpress);
            } else if (event.getAction() == MotionEvent.ACTION_UP) {
                move_Up.setImageResource(R.drawable.backward);
                move_Up(currently_Music.getText().toString());
                lrc_time = new ArrayList<String>();
                lrc_word = new ArrayList<String>();
                showLrc(currently_Music.getText().toString());// 歌词显示
            }
            return false;
        }
};
```

❑ 定义方法move_Down()和move_Up()，分别用于播放下一首歌曲和上一首歌曲，对应代码如下所示。

```
private void move_Down(String musicName) {
	for (int i = 0; i < music_List.size(); i++) {
		if (musicName.equals(music_List.get(i))) {
			if ((i + 1) < music_List.size()) {
				selectName = music_List.get(i + 1);
				play_Music();
				return;
			} else {
				selectName = music_List.get(0);
				play_Music();
				return;
			}
		}
	}
}

private void move_Up(String musicName) {
	for (int i = 0; i < music_List.size(); i++) {
		if (musicName.equals(music_List.get(i))) {
			if ((i - 1) >= 0) {
				selectName = music_List.get(i - 1);//移动到上一首歌曲
				play_Music();
				return;
			} else {
				selectName = music_List.get(music_List.size() - 1);
				play_Music();
				return;
			}
		}
	}
}
```

❑ 定义方法play_Mode()来设置系统的播放模式，本实例有单曲循环、顺序播放和随机播放3种模式。对应代码如下所示。

```
private void play_Mode() {
	if ("is_Sigle".equals(play_Mode)) {//单曲循环
		play_Music();
	}
	if ("is_Order".equals(play_Mode)) {//顺序播放
		move_Down(currently_Music.getText().toString());
	}
	if ("is_Random".equals(play_Mode)) {//随机播放
		Random r = new Random();
		int idx = r.nextInt(music_List.size());//随机生成 [0,music_List.size())的 INT 值
		selectName = music_List.get(idx);
		play_Music();
	}
}
```

❑ 定义方法play_Music()播放指定的音乐文件，对应代码如下所示。

```
private void play_Music() {
```

```
        try {

            mplayer.reset();
            mplayer.setDataSource(query());//文件流中选择歌曲
            mplayer.prepare();
            mplayer.start();
            currently_Music.setText(selectName);
            /*
            if(cursor.moveToFirst()){
                String title=systemProvider.getArtist();
                currently_Music.setText(title);
            }*/

            seekBar.setMax(mplayer.getDuration());//音频文件持续时间
            seekBar.setProgress(1);
            currently_Time.setText(getFileTime(mplayer.getCurrentPosition()));
        //  lrcTime.setText(systemProvider.getArtist());
            handler.removeCallbacks(thread_One);
            end_Time.setText(getFileTime(mplayer.getDuration()));
            handler.postDelayed(thread_One, 1000);
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```

- 定义方法query()查询歌曲路径，对应代码如下所示。

```
public String query() {
    ContentResolver cr = getContentResolver();
    Uri uri = DBProvider.CONTENT_URI;
    String[ ] projection = { "path" };
    String selection = "fileName=?";
    String[ ] selectionArgs = { selectName };
    Cursor c = cr.query(uri, projection, selection, selectionArgs, null);
    if (c.moveToFirst()) {
        String path = c.getString(0);
        return path;
    }
    return null;
}
```

- 定义方法getFileTime()获取音乐文件的播放持续时间长的格式化字符串，其返回值是一个格式化时间字符串。对应代码如下所示。

```
private String getFileTime(int timeMs) {
    int totalSeconds = timeMs / 1000;//获取文件有多少秒
    StringBuilder mFormatBuilder = new StringBuilder();
    Formatter mFormatter = new Formatter(mFormatBuilder, Locale
                .getDefault());
    int seconds = totalSeconds % 60;
    int minutes = (totalSeconds / 60) % 60;
    int hours = totalSeconds / 3600;
    mFormatBuilder.setLength(0);
```

```
        if (hours > 0) {
            return mFormatter.format("%d:%02d:%02d", hours, minutes, seconds)
                    .toString();//格式化字符串
        } else {
            return mFormatter.format("%02d:%02d", minutes, seconds).toString();
        }
}
```

另外在文件 MainPlayActivityRoot.java 中，定义了主界面中控制按钮的响应方法，其主要实现代码如下所示。

```
/**
 * 清单按钮
 */
protected ImageButton list_bt;
/**
 * 返回按钮
 */
protected ImageButton back_bt;
/**
* 停止按钮
*/
protected ImageButton stop_bt;
/**
* 播放按钮
*/
protected ImageButton play_bt;
/**
* 上一首歌曲
*/
protected ImageButton move_Up;
/**
* 下一首歌曲
*/
protected ImageButton move_Down;
/**
* 歌曲结束时间
*/
protected TextView end_Time;
/**
* 当前播放时间
*/
protected TextView currently_Time;
/**
* 当前播放时间
*/
protected TextView currently_Music;
/**
* 音轨
*/
protected SeekBar seekBar;
/**
* 歌词显示
*/
```

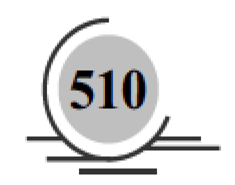
```
protected TextView lrcTime;
/**
* 播放器是否停止
*/
protected boolean is_stopping = false;
/**
* 系统自带播放器控件
*/
protected MediaPlayer mplayer;
/**
* 选中的歌曲
*/
protected String selectName;
/**
* 正在播放的歌曲
*/
protected String playingName;
/**
*  播放模式：默认为随机播放模式
*/
protected String play_Mode = "is_Random";
/**
* 歌词显示模式
*/
protected String lrc_Show;
/**
* 歌曲列表
*/
protected List<String> music_List = new ArrayList<String>();
/**
* 线程
*/
protected Handler handler;
```

16.4.3 播放列表功能

系统的歌曲列表界面如图 16-10 所示。



图 16-10 歌曲列表界面



（1）编写布局文件 play_list.xml，主要代码如下所示。

```
<LinearLayout android:layout_width="fill_parent"
        android:gravity="center" android:layout_height="wrap_content"
        android:background="@drawable/footer_bar">
        <TextView android:text="歌曲列表" android:id="@+id/music_list"
                android:textSize="@dimen/music_list_title" android:textStyle="bold"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"></TextView>

</LinearLayout>

<ListView android:id="@+id/show_play_list"
        android:layout_width="fill_parent" android:layout_height="337px"></ListView>
<LinearLayout android:layout_width="fill_parent"
        android:gravity="right" android:layout_height="wrap_content"
        android:background="@drawable/footer_bar">
        <ImageButton android:id="@+id/back" android:background="@drawable/back"
                android:layout_width="wrap_content" android:layout_height="wrap_content"></ImageButton>
</LinearLayout>
```

在 Android 系统中有一个名为 ListView 的视图，其特点是一个有 BaseAdapter 的属性，从上到下，或从左到右的显示方式。系统默认的方式每一行只显示一个 TextView，本播放列表实现了自定义的方式，用 ListView 的每一行显示一个音乐图片和一个歌曲名字。在此定义了类 MusicAdapter 来继承 BaseAdapter，然后通过算法对这个适配器进行扩展，扩展成为第一行能显示一张图片和一个歌曲名字。由于 BaseAdapter 是一个抽象类，我们需要实现里面的抽象方法 getView()，该方法返回一个 View，即视图。视图可以显示在 Activity 上，所以就可以看到我们想要的歌曲列表界面。

ListView 同样有一个监听器 new onItemClickListener()，我们只要实现这个方法就可以监听点击事件，当点击到每一行时，可以通过 ListView.getItemAtPositon(int position)得到该行上的信息。这样就可以通过 Intent 将数据传入到其他的 Activity。本程序的思路是当点击一行，会跳转到另一个 Activity 里面，这个 Activity 和歌曲列表类似，也是一个 ListView，该界面将在 16.4.4 节介绍。

歌曲列表是从播放主界面跳转过来的，能跳到该歌曲列表的前提是数据库中有歌曲列表的存在。因为每次歌曲列表显示时会查询数据库中的歌曲列表。如果不存在会提示是空列表，选择到 SDCard 中添加歌曲，如图 16-11 所示。



图 16-11　列表为空时的提示

（2）编写程序文件 PlayListActivity.java，首先定义方法 setListener()来监听用户的选择操作，将用户选择的歌曲进行播放；然后定义方法 query()来查询系统库内的歌曲，如果为空则显示“播放列表为空”；最后定义方法 showDialog()来显示图 16-11 所示的提示框。文件 PlayListActivity.java 的主要代码如下所示。

```
private void setListener(){
        playlist.setOnItemClickListener(new OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                                long arg3) {
```

```
                Intent intent = new Intent(PlayListActivity.this, Menu.class);
                selectName = playlist.getItemAtPosition(arg2).toString();
                startActivityForResult(intent, 2);
            }
        });
        back.setOnTouchListener(new OnTouchListener() {

            @Override
            public boolean onTouch(View v, MotionEvent event) {

                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    v.setBackgroundResource(R.drawable.back_pressed);
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    v.setBackgroundResource(R.drawable.back);
                    Intent intent = new Intent();
                    intent.setClass(PlayListActivity.this, MainPlayActivity.class);
                setResult(0, intent);
                finish();
                }

                return false;
            }
        });
    }

    @Override
    protected void onPause() {

        super.onPause();
        Log.v("log", "playListActivity is in pause state");
        SharedPreferences sp=getSharedPreferences("MUSIC", MODE_WORLD_WRITEABLE);
        SharedPreferences.Editor editor=sp.edit();


        editor.putString("SELECTNAME", selectName);
        String str=StringHelper.toStringAll(list);
        editor.putString("MUSIC_LIST", str);
        editor.commit();
    }
    public String[ ] query() {//查询数据库
        cr = getContentResolver();
        uri = DBProvider.CONTENT_URI;
        list.clear();
        String[ ] projection = { "filename", "path" };
        Cursor c = cr.query(uri, projection, null, null, null);
        if (c.getCount() == 0) {
            showDialog("播放列表为空");
        }
        String[ ] music = new String[c.getCount()];
        if (c.moveToFirst()) {
            for (int i = 0; i < c.getCount(); i++) {
```

```
                c.moveToPosition(i);
                String filename = c.getString(0);
                music[i] = filename;
                list.add(filename);
            }
        }
        if (music.length > 0) {
            playlist.setAdapter(new MusicAdapter(this, list));
        }
        return music;
    }

    private void showDialog(String msg) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(msg).setCancelable(false).setPositiveButton("从 SD 卡",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        Intent intent = new Intent(PlayListActivity.this,
                                FileExplorerActivity.class);
                    //  startActivity(intent);
                        startActivityForResult(intent, 2);
                    }
                }).setNegativeButton("取消", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Intent intent = new Intent();
            setResult(0, intent);
            finish();
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }
```

16.4.4　菜单功能模块

本系统实例的菜单功能界面如图 16-12 所示。

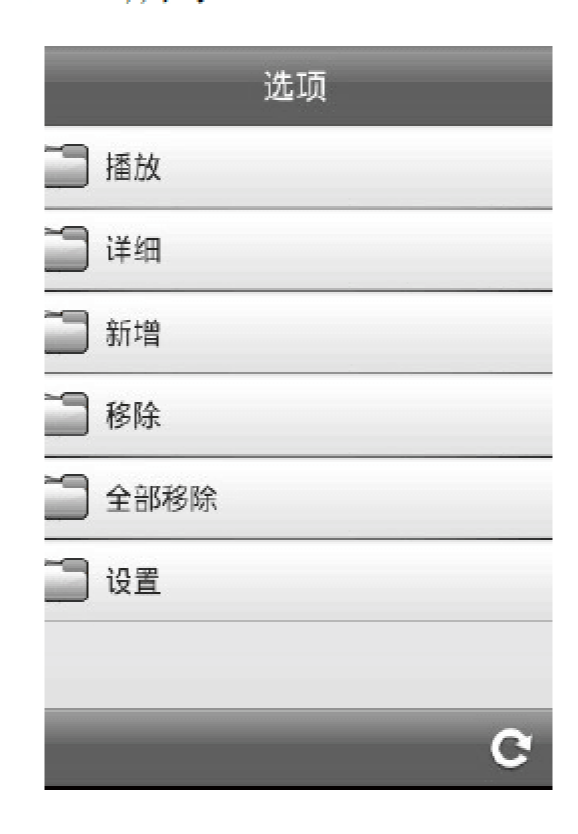


图 16-12　菜单选项界面

（1）编写布局文件 menu.xml，主要代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:background="@drawable/list_bg">
    <LinearLayout android:layout_width="fill_parent"
        android:gravity="center" android:layout_height="wrap_content"
        android:background="@drawable/footer_bar">
        <TextView android:text="选项" android:id="@+id/select_item"
            android:textSize="@dimen/music_list_title" android:textStyle="bold"
            android:layout_width="wrap_content" android:layout_height="wrap_content"></TextView>
    </LinearLayout>
    <ListView android:id="@+id/menu" android:layout_width="wrap_content"
        android:background="@drawable/list_item_bg"
        android:layout_height="wrap_content"></ListView>
    <TextView android:layout_width="wrap_content"
        android:layout_height="50px"></TextView>
    <LinearLayout android:layout_width="fill_parent" android:gravity="right"
        android:layout_height="wrap_content"
        android:background="@drawable/footer_bar">
        <ImageButton android:id="@+id/back"
            android:background="@drawable/back"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"></ImageButton>
    </LinearLayout>
```

（2）编写文件 menu.java，其具体实现流程如下。

- 使用List<String>容器保存String类型的字符，保存了“播放”、“新增”、“详细”、“移除”、“全部移除”和“设置”等选项。对应代码如下所示。

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.menu);
    menuLV = (ListView) findViewById(R.id.menu);
    back=(ImageButton)findViewById(R.id.back);
    select_Item = (TextView) findViewById(R.id.select_item);
    select_Item.setTextColor(Color.WHITE);
    SharedPreferences sp = getSharedPreferences("MUSIC",
            MODE_WORLD_READABLE);
    if (sp != null) {

        selectName=sp.getString("SELECTNAME", null);
    }

    List<String> seclect_items = new ArrayList<String>();
    seclect_items.add("播放");
    seclect_items.add("详细");
    seclect_items.add("新增");
    seclect_items.add("移除");
    seclect_items.add("全部移除");
    seclect_items.add("设置");
    menuLV.setAdapter(new MusicAdapter(this, seclect_items));

    back.setOnTouchListener(backListener)
    ;
```

❑ 使用case语句根据用户选择的选项来到对应的界面，对应代码如下所示。

```
    menuLV.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                long arg3) {
            SharedPreferences sp=getSharedPreferences("MUSIC", MODE_WORLD_WRITEABLE);
            SharedPreferences.Editor editor=sp.edit();
            switch (arg2) {
            case 0:// 播放
                Bundle bundle = new Bundle();
            bundle.putInt("operate", 0);
            Intent intent = new Intent();
            intent.putExtras(bundle);
            setResult(2, intent);
            finish();
                break;
            case 2:
                Intent intent_add = new Intent(Menu.this,
                        FileExplorerActivity.class);
                editor.putString("SELECTNAME", null);
                editor.commit();
            //  startActivity(intent_add);
                startActivityForResult(intent_add, 3);
                break;
            case 3://移除
                showDialog(selectName);
                break;
            case 4://全部移除
                showDialog("");
                break;
            case 5://设置
                Intent intent_set = new Intent(Menu.this, PlaySetting.class);
                editor.putString("SELECTNAME", null);
                editor.commit();
                startActivityForResult(intent_set, 3);
                break;
            default:
                break;
            }
        }
    });
}
```

❑ 定义方法showDialog()，用于提示用户确认是否移除或全部移除列表中的音频文件。对应代码如下所示。

```
private void showDialog(final String msg) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    if (msg.equals(""))
        builder.setMessage("全部移除?");
    else
        builder.setMessage("是否移除?");
    builder.setCancelable(false).setPositiveButton("是",
```

```
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        if (msg.equals(""))
                            del_All();
                        else
                            del_One(selectName);
                        Intent intent = new Intent();
                    setResult(6, intent);
                    finish();
                    }
                }).setNegativeButton("否", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
}
```

❑ 定义方法del_One()来删除单首歌曲，定义方法del_All()来删除全部歌曲。对应代码如下所示。

```
private void del_One(String musicName) {//删除单首歌曲
        ContentResolver cr = getContentResolver();
        Uri uri = DBProvider.CONTENT_URI;
        String where = "fileName=?";
        String[ ] selectionArgs = { musicName };
        cr.delete(uri, where, selectionArgs);
}

private void del_All() {//删除全部歌曲
        ContentResolver cr = getContentResolver();
        Uri uri = DBProvider.CONTENT_URI;
        cr.delete(uri, null, null);
}
```

16.4.5 播放设置界面

本系统的播放设置界面如图 16-13 所示。



图 16-13　播放设置界面

（1）编写布局文件 setting.xml，主要代码如下所示。

```
<LinearLayout android:layout_width="fill_parent"
    android:gravity="center" android:layout_height="wrap_content"
    android:background="@drawable/footer_bar">
    <TextView android:text="设定" android:id="@+id/setting"
        android:textSize="@dimen/music_list_title" android:layout_width="wrap_content"
        android:layout_height="wrap_content"></TextView>
</LinearLayout>
<LinearLayout android:orientation="horizontal"
    android:layout_width="wrap_content" android:gravity="center_vertical"
    android:layout_height="wrap_content">
    <TextView android:text="播放模式" android:id="@+id/setting"
        android:textSize="@dimen/text_size" android:layout_width="fill_parent"
        android:layout_height="wrap_content"></TextView>
    <RadioGroup android:id="@+id/RadioGroup"
        android:layout_width="wrap_content" android:layout_height="wrap_content">
        <RadioButton android:text="单曲循环" android:id="@+id/sigle_play"
            android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
        <RadioButton android:text="顺序播放" android:id="@+id/order_play"
            android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
        <RadioButton android:text="随机播放" android:id="@+id/random_play"
            android:checked="true" android:layout_width="wrap_content"
            android:layout_height="wrap_content"></RadioButton>
    </RadioGroup>
</LinearLayout>
<LinearLayout android:orientation="horizontal"
    android:layout_width="wrap_content" android:gravity="center_vertical"
    android:layout_height="wrap_content">
    <TextView android:text="歌词显示" android:id="@+id/setting"
        android:textSize="@dimen/text_size" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <ToggleButton android:text="" android:id="@+id/ly_lrc"
        android:checked="false" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
<TextView android:layout_width="fill_parent"
    android:layout_height="150px"></TextView>
<AbsoluteLayout android:background="@drawable/footer_bar"
    android:layout_width="fill_parent" android:layout_height="wrap_content">
    <ImageButton android:id="@+id/make" android:background="@drawable/share"
        android:layout_x="270px" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <ImageButton android:id="@+id/cancel" android:layout_x="5px"
        android:background="@drawable/back" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</AbsoluteLayout>
```

（2）编写程序文件 PlaySetting.java，其主要功能如下所示。

❑ 设置播放模式

在设置播放模式时用的是 RadioGroup 组件，这个组件有单项选择的功能，里面有 RadioButton 项，

多个 RadioButton 项只能同时选中一个，该播放器播放模式有单曲循环、随机播放、顺序播放等功能。MediaPlayer 有一个监听器，它监听着歌曲是否正在播放或者是否播放完成，当歌曲播放完成时，会触发方法 OnCompletionListener()，在该方法里面可以处理歌曲播放完成后的操作。RadioGroup 可以进行单项选择操作。

❑ 歌词设置

歌词是否显示是一个开关按钮 ToggleButton 实现的，有 ON 和 OFF 状态，当为 ON 时显示歌词，为 OFF 时关闭歌词。

ToggleButton 同样也有一个监听器，可以获得 ToggleButton 的不同状态。使用前对它进行实例化 (ToggleButton) View.findViewById(R.id.ly_lrc);并且用 ToggleButton.isChecked();获得开关状态。播放模式状态和歌词显示状态的操作结果都将以一个标志，被写在一个配置文件中，这是关于 Android 的存储方式。

了解文件 PlaySetting.java 的实现原理和功能后，其实现代码就非常容易理解了，主要代码如下所示。

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.setting);
        set_textView = (TextView) findViewById(R.id.setting);
        set_textView.setTextColor(Color.WHITE);
        sigle_Play = (RadioButton) findViewById(R.id.sigle_play);
        order_Play = (RadioButton) findViewById(R.id.order_play);
        random_Play = (RadioButton) findViewById(R.id.random_play);
        lyLrc = (ToggleButton) findViewById(R.id.ly_lrc);
        set_bt = (ImageButton) findViewById(R.id.make);
        cancel_bt = (ImageButton) findViewById(R.id.cancel);
        set_bt.setOnTouchListener(setting_bt_Listener);
        cancel_bt.setOnTouchListener(cancel_bt_Listener);
}

OnTouchListener setting_bt_Listener = new OnTouchListener() {//设置确定按钮监听器
        @Override
        public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                        v.setBackgroundResource(R.drawable.share_pressed);
                        SharedPreferences sp = getSharedPreferences("SET_MSG",
                                        MODE_WORLD_WRITEABLE);// 文件共享
                        SharedPreferences.Editor editor = sp.edit();
                        if (sigle_Play.isChecked()) {
                                editor.putString("sigle_Play", "is_Sigle");
                                editor.putString("order_Play", null);
                                editor.putString("random_Play", null);
                        }
                        if (order_Play.isChecked()) {
                                editor.putString("sigle_Play", null);
                                editor.putString("order_Play", "is_Order");
                                editor.putString("random_Play", null);
                        }
                        if (random_Play.isChecked()) {
                                editor.putString("sigle_Play", null);
                                editor.putString("order_Play", null);
```

```
                editor.putString("random_Play", "is_Random");
            }
            if (lyLrc.isChecked()) {
                editor.putString("lyLrc", "is_Show");
            }
            if (!lyLrc.isChecked()) {
                editor.putString("lyLrc", null);
            }
            editor.commit();// 提交
            Intent intent = new Intent();
            setResult(4, intent);
        finish();
        } else if (event.getAction() == MotionEvent.ACTION_UP) {
            v.setBackgroundResource(R.drawable.share);
        }
        return false;
    }
};

OnTouchListener cancel_bt_Listener = new OnTouchListener() {// 取消监听器
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            v.setBackgroundResource(R.drawable.back_pressed);
        } else if (event.getAction() == MotionEvent.ACTION_UP) {
            v.setBackgroundResource(R.drawable.back);
            Intent intent = new Intent();
            setResult(3, intent);
        finish();
        }
        return false;
    }
};
```

16.4.6　设置显示歌词

显示歌词功能比较重要，所以笔者决定单独讲解其实现原理。本播放器的歌词是.Lrc 格式文件，查看.Lrc 文件中的歌词格式如下所示。

```
[00:16.18]我爱你 心爱的姑娘
```

歌词格式是以“时间+歌词”的格式存储。

接下来将介绍如何将.Lrc 中的歌词读取出来并存储在 Android 的配置文件中。

（1）用 XML 配置文件的存储

在 Android 系统中，SD 卡的目录结构如图 16-14 所示。

从图 16-14 中可以看到一个名为 SDCard 的目录，该目录即为扩展卡，里面预先存放着音频文件和.Lrc 歌词文件，我们定义如下代码来指定.Lrc 文件存在的路径，并将文件读取到 BufferReader 中。

```
BufferedReader buffer=new BufferedReader(new FileReader(new File("/sdcard/"+ musicName + ".lrc")));
```

由于我们要分别存放时间和歌词，所以应该定义两个 List<String>容器来存放时间和歌词。在读取.Lrc 文件时，每次读取一行，再用算法将时间和歌词分开后放到一个数组里面，并分别存放在两个

List 中。由于歌曲在播放时会存在界面之间的跳转，所以歌词必须固定存放在一个文件中，而不能作为一个对象，因此，我们将两个时间 List 和歌词 List 再写进一个配置文件中。

Android 为我们提供了共享文件类 SharedPreferences，它有一个方法 getSharedPreferences(参数 1，参数 2)，参数 1 表示写进时的标记，便于在从其中读取出来时的标记；参数 2 表示读取模式，有只写模式（MODE_WORLD_WRITEABLE）和只读模式（MODE_WORLD_READABLE），在写之前将其设置为编辑状态，下面是使用静态方法设置编辑状态的代码。

```
SharedPreferences.Editor editor = sp.edit();
```

然后在对象 editor 中可以存入一个 HashMap<key,values>类型的键值，其格式是 putString(KEY, VALUES)，这样就可以将 List 中的对象转化成一样长的字符放进配置文件中。

当写入成功时，Android 系统会自动在目录 data/data/工程包名/shared_prefs/中生成一个配置文件，如图 16-15 所示。

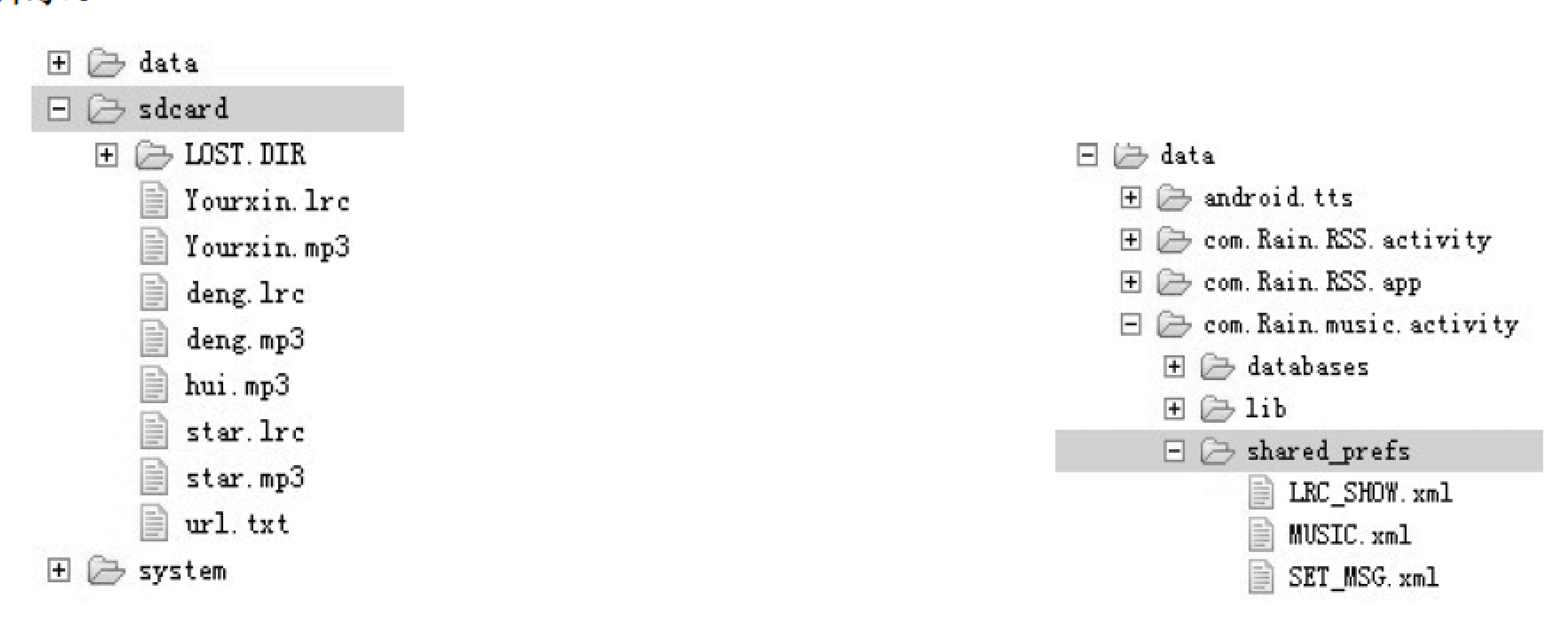


图 16-14　SD 卡的目录结构　　　　图 16-15　配置文件

打开播放模式的 XML 配置文件，在此文件中是以 map 的形式存储的，键名是<string name="random_Play"></string>，其值是 is_Radom，如图 16-16 所示。

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <map>
    <string name="random_Play">is_Random</string>
    <string name="lyLrc">is_Show</string>
    <null name="order_Play" />
    <null name="sigle_Play" />
  </map>
```

图 16-16　XML 配置文件

（2）读取 XML 配置文件

我们仍以播放模式读取为例，当需要用到播放模式的确定时将读取.XML 文件，同样用共享文件类 SharedPreferences 通过用方法 getSharedPreferences("SET_MSG",MODE_WORLD_READABLE)，并且是只读方式获得.XML 的文件内容。SharedPreferences 的对象调用方法 getString("sigle_Play", null)，此方法会返回一个 String 类型的值，即是我们以前存储进去的 String 值。当该标记不存在时，会默认返回一个 null 值。读取 XML 配置文件成功后，就可以根据配置的值对程序进行操作了。

16.4.7　文件浏览器模块

本项目程序实现了文件浏览器的功能，作为一个文件浏览器应该具有浏览功能。当程序运行到浏览界面时，会有各文件的目录显示及图标标识。从文件浏览器中我们能看到各文件，而且能对其进行



操作，本程序是专为播放器添加歌曲而设计的，因此功能仅限于对媒体文件的浏览，以及含有媒体文件的目录的浏览，所以功能比较局限。

当显示菜单界面时，通过新增选项进入到文件浏览器中，或者当播放列表为空时，会提示进入文件浏览器进行歌曲新增操作。其中文件浏览器界面如图 16-17 所示，SD 卡目录界面如图 16-18 所示。

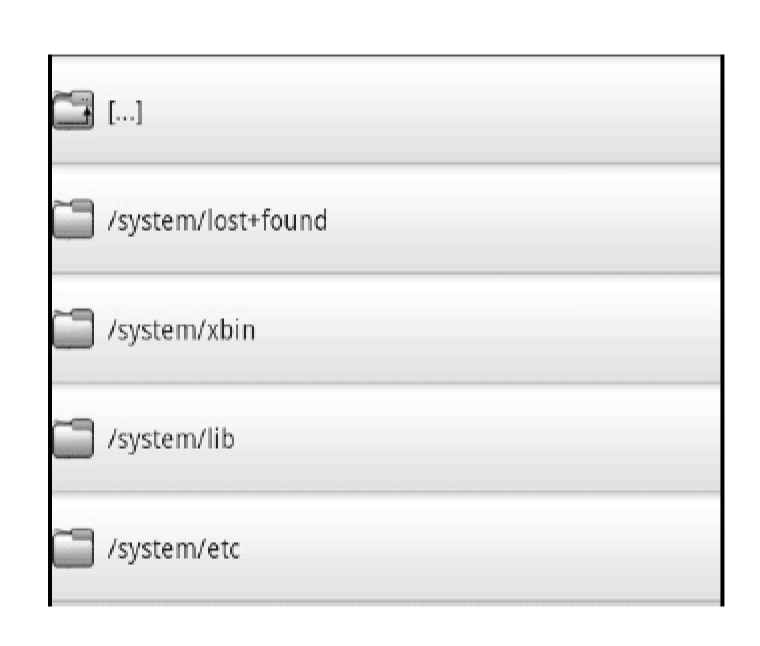



图 16-17　文件浏览器界面



图 16-18　SD 卡目录界面

文件浏览器界面布局格式类似前面介绍的菜单，只是在界面的第一行新增了一个返回根目录的功能。由于程序只关系到目录/sdcard 下的文件，所以用程序屏蔽了其他的目录，这里只显示两个目录“/sdcard”和“/system”。播放器只需要用到媒体文件，所以代码也屏蔽了其他文件的子目录。当选中 sdcard 会进入到图 16-18，该目录下只显示媒体文件，如.MP3 和 SD 卡下的子目录。选中 system 会进入到图 16-17，该目录会显示 system 下的各级子目录。当有媒体文件时才会出现添加 Dialog。

当要添加选中的歌曲时，程序有自动判断功能，首先弹出 Dialog。单击“确定”按钮后，程序会查询数据库中的歌曲，调用方法 query(fileName)，根据歌曲名字查询，如果歌曲不存在，则调用方法 insertMusic(file)，如果该歌曲名字已经存在则弹出 Dialog 对话框。

（1）编写文件 directory_list.xml，实现文件浏览界面的布局，主要代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:background="@drawable/list_bg">
    <LinearLayout android:layout_width="fill_parent"
        android:gravity="center" android:layout_height="wrap_content"
        android:background="@drawable/footer_bar">
        <TextView android:text="SD 卡" android:id="@+id/store_card" android:textStyle="bold"
            android:layout_width="wrap_content" android:layout_height="wrap_content"
            android:textSize="@dimen/music_list_title"></TextView>
    </LinearLayout>

    <ListView android:id="@id/android:list" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
        <TextView   android:layout_width="10px"
        android:layout_height="wrap_content" />
    <TextView android:id="@id/android:empty" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="@string/no_files"
/>
```

（2）编写文件 FileExplorerActivity.java，在此定义了文件浏览器类 FileExplorerActivity，此类继承了 ListActivity，此 Activity 是一个 ListView 界面。整个界面是一个 ListView 布局，而每一行是一个 LinearLayout 水平方式布局，上面将放置一个图片和一个文件全路径。该文件全路径被存放到数据库中，以

便歌曲播放能查询到歌曲路径源。

文件 FileExplorerActivity.java 的主要代码如下所示。

```
public class FileExplorerActivity extends ListActivity {
    private List<String> items = null;
    private TextView store_Card;
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.directory_list);
        store_Card = (TextView) findViewById(R.id.store_card);
        store_Card.setTextColor(Color.WHITE);
        fill(new File("/").listFiles());
    }

    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        int selectionRowID = (int) position;
        File file = new File(items.get(selectionRowID));
        if (selectionRowID == 0) {
            fillWithRoot();
        } else {
            if (file.isDirectory()) {
                fill(file.listFiles());
            } else {
                Intent intent = this.getIntent();
                intent.putExtra("filePath", file);
                FileExplorerActivity.this.setResult(0, intent);
                showDialog("加入播放列表?", file);
            }
        }
    }

    private void fillWithRoot() {
        fill(new File("/").listFiles());
    }

    private void fill(File[ ] files) {
        items = new ArrayList<String>();
        items.add(getString(R.string.to_top));
        for (File file : files) {
            if (file.isDirectory()) {
                if ((file.getPath().indexOf("/sdcard")) != -1
                        || (file.getPath().indexOf("/system")) != -1)
                    items.add(file.getPath());
            }
            if ((file.getPath().indexOf(".mp3")) != -1) {
                items.add(file.getPath());
            }
        }
        setListAdapter(new MusicAdapter(this, items));
```

```
}

private void showDialog(String msg, final File file) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(msg).setCancelable(false).setPositiveButton("确定",
                        new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int id) {
                                        String fileName = file.getName().substring(0,
                                                        file.getName().indexOf("."));
                                        Log.i("info", fileName);
                                        if (query(fileName)) {
                                                insertMusic(file);//添加音乐
                                        }
                                }
                        }).setNegativeButton("取消", null);
        AlertDialog alert = builder.create();
        alert.show();
}

// 添加音乐到播放列表
private final void insertMusic(File file) {
        ContentResolver cr = getContentResolver();
        ContentValues values = new ContentValues();
        Uri uri = DBProvider.CONTENT_URI;
        String fileName = file.getName().substring(0,
                        file.getName().indexOf("."));
        values.put(FileColumn.NAME, fileName);
        values.put(FileColumn.PATH, file.getAbsolutePath());
        values.put(FileColumn.TYPE, "Music");
        values.put(FileColumn.SORT, "popular");
        cr.insert(uri, values);
        Toast.makeText(FileExplorerActivity.this, "已加入", Toast.LENGTH_LONG)
                        .show();
        Intent intent = new Intent();
setResult(6, intent);
finish();
}

private boolean query(String name) {
        ContentResolver cr = getContentResolver();
        Uri uri = DBProvider.CONTENT_URI;
        String[ ] projection = { "filename" };
        Cursor c = cr.query(uri, projection, null, null, null);
        if (c.moveToFirst()) {
                for (int i = 0; i < c.getCount(); i++) {
                        c.moveToPosition(i);
                        String filename_DB = c.getString(0);
                        if (name.equals(filename_DB)) {//判断播放列表中是否存在该歌曲
                                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                                builder.setMessage("文件已存在").setCancelable(false)
                                                .setPositiveButton("返回列表",
```

```
                                        new DialogInterface.OnClickListener() {
                                            public void onClick(
                                                    DialogInterface dialog, int id) {
                                                Intent intent = new Intent(
                                                        FileExplorerActivity.this,
                                                        PlayListActivity.class);
                                                startActivity(intent);
                                            }
                                        }).setNegativeButton("重新添加", null);
                        AlertDialog alert = builder.create();
                        alert.show();
                        return false;
                    }
                }
            }
            return true;
        }
```

上述 ListView 实现了自动判断的功能，即程序可以通过访问扩展卡中的文件属性而自动识别文件属性。当为一个 MP3 格式文件时，则前面图标显示 MP3 图标，当为一个文件目录时，则图标标识为一个文件。文件浏览器是用递归算法实现的，其中 fillWithRoot()用于返回根目录的列表。

16.4.8 数据存储

在播放器正常运行时，由于各界面存在相互跳转，为了避免数据在界面跳转的过程中丢失，我们需要将一些数据进行临时存储或者永久存储。

Android 作为一种手机操作系统，提供了 Preference（配置）、File（文件）、SQLite 数据和网络等存取数据的方式。

另外，在 Android 中各个应用程序组件之间是相互独立的，彼此的数据不能共享。为了实现数据的共享，Android 提供了 Content Provider 组件来实现应用程序之间数据的共享。

（1）SharedPreferences

SharedPreference 提供了一种轻量级的数据存取方法，通常用于存储数据比较少的信息或一些简单的配置信息。它以“键-值”（是一个 Map）对的方式，将数据保存在一个 XML 配置文件中。

在本实例中用到了如下两种数据存储接口。

- android.content.SharedPreferences：提供了保存数据的方法。
- android.content.SharedPreferences.Editor：提供了获得数据的方法。

注意：有关上述SharedPreferences数据存储的知识，请读者参阅相关资料，这并不是本书的重点。

以播放器中的播放模式存取为例，实现流程如下。

- XML配置文件的读取

仍以播放模式读取为例，当需要用到播放模式的确定时，我们将读取.XML 文件，同样用共享文件类 SharedPreferences，通过用方法 getSharedPreferences("SET_MSG",MODE_WORLD_READABLE)，并且是只读方式获得 XML 的文件内容。SharedPreferences 的对象调用方法 getString("sigle_Play", null)，方法返回一个 String 类型的值，即是我们以前存储进去的 String 值。此方法当该标记不存在时会默认

返回一个 null 值。获得成功后我们就可以运用当前的值再对程序进行操作了。

- XML配置文件的存储

在类 SharedPreferences 中有一个方法 getSharedPreferences(参数 1，参数 2)，参数 1 为写进时的标记，便于在从其中读取出来时的标记，参数 2 为读取模式，有只写模式（MODE_WORLD_WRITEABLE）和只读模式（MODE_WORLD_READABLE），在写之前将其置入编辑状态，用静态方法 SharedPreferences.Editor editor = sp.edit(); 然后对象 editor 可以存入一个 HashMap<key,values>类型的键值，即 putString(KEY, VALUES)，这样，我们可以将 List 中的对象转化成一样长的字符放进配置文件中。当写入成功时，Android 系统会自动在目录 data/data/工程包名/shared_prefs 下生成一个配置文件。

（2）File 存储方式

我们可以将一些数据直接以文件的形式保存在设备中。例如一些文本文件、PDF 文件、音视频文件和图片等。Android 提供了如下文件读写的方法。

- Context.openFileInput()：获得标准Java文件输入流（FileInputStream）。
- Context.openFileOutput()：获得标准Java文件输出流（FileOutputStream）。
- Resources.openRawResource (R.raw.myDataFile)：返回InputStream。

（3）SQLiteDatabase 数据库

SQLite 是一个嵌入式数据库引擎，针对内存等资源有限的设备（如手机、PDA、MP3）提供的一种高效的数据库引擎。SQLite 数据库不像其他的数据库（例如 Oracle），它没有服务器进程，所有的内容包含在同一个单文件中，该文件是跨平台的可以自由复制。基于其自身的先天优势，SQLite 在嵌入式领域得到了广泛应用。

- SQLiteDatabase类：代表一个数据库对象，在里面提供了操作数据库的一些方法，具体方法请读者参考相关教材或书籍。
- SQLiteOpenHelper类：是SQLiteDatabase的一个帮助类，用来管理数据库的创建和版本更新。一般的用法是定义一个类继承之，并实现其两个抽象方法onCreate(SQLiteDatabase db)和onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)来创建和更新数据库。

Android 的 3 种数据存储方式则让我们可以轻松方便地进行程序编写和数据的访问，更不会让不该消失的数据消失，这对我们进行程序书写有很大的帮助。

第 17 章　移动阅读器系统

RSS 是在线共享内容的一种简易方式（也叫聚合内容，Really Simple Syndication）。通常在时效性比较强的内容上使用 RSS 订阅能更快速获取信息，网站提供 RSS 输出，有利于让用户获取网站内容的最新更新。在本书前面的内容中，已经讲解过一个简单 RSS 系统的实现流程，本章将通过一个综合实例的实现过程，详细讲解在 Android 手机中开发一个 RSS 阅读器的方法。

17.1　实 现 流 程

知识点讲解：光盘:视频\知识点\第 17 章\实现流程.avi

本项目实例的功能是，在手机中显示指定 RSS 的信息，即设置显示网易博客 http://woshiyigebing12345.blog.163.com 用户的日志信息。

本项目的具体实现流程如图 17-1 所示。

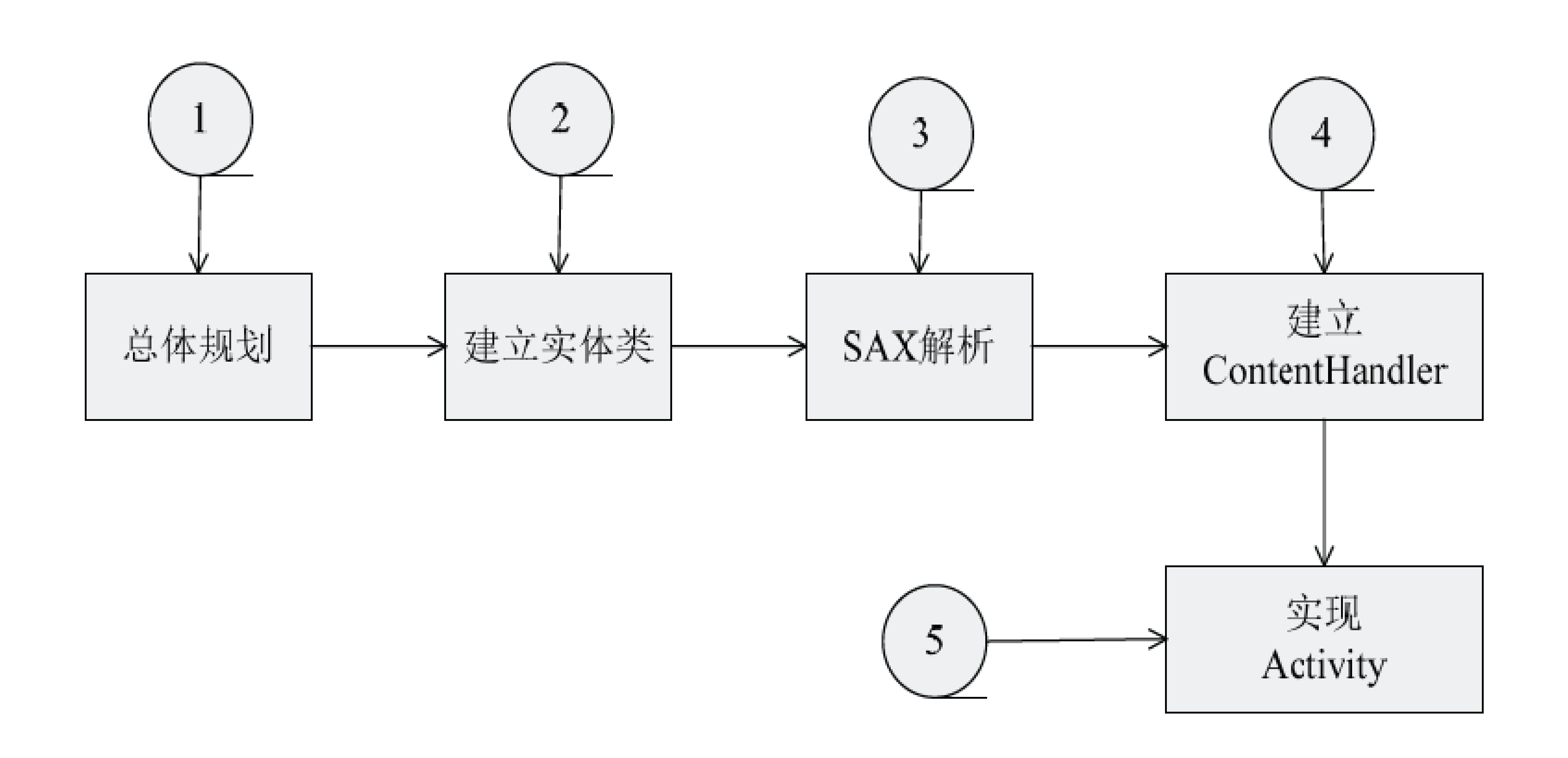


图 17-1　实现流程图

17.2　具 体 实 现

知识点讲解：光盘:视频\知识点\第 17 章\具体实现.avi

从本节内容开始，将着重介绍本项目的具体实现过程，以及各个代码的具体实现过程，并讲解其中的技巧和要点，使读者的水平更上一层楼。

实例	功能	源码路径
实例 17-1	开发一个 RSS 阅读器	光盘:\codes\17\RSSREAD

17.2.1　建立实体类

一个 RSS 文件可以被认为是由一个 RSS 的一些描述性信息和里面的 item 元素组成的，例如关于 RSS 的描述性信息如下。

- ❑ title：标题信息。
- ❑ link：链接信息。
- ❑ description：描述信息。

item 里的信息如下。

- ❑ title：标题信息。
- ❑ link：链接信息。
- ❑ description：描述信息。
- ❑ pubDate：发布的日期。

在本项目实例中需要建立如下两个实体类。

- ❑ RSSFeed：用于和一个RSS的完整XML文件相对应。
- ❑ RSSItem：用于和一个RSS中Item标签相对应。

在解析 RSS 文件时，可以将文件里的信息解析出来放到实体类里，这样就可以直接操作该实体类了。下面开始讲解上述两个实体类的具体实现过程。

1. RSSFeed 类

RSSFeed 类的功能是建立和一个完整 XML 文件的对应，其中方法 addItem()用于将一个 RSSItem 添加到 RSSFeed 类里；方法 getAllItemsForListView()负责从 RSSFeed 类里生成 ListView 列表所需要的数据。RSSFeed 类的具体实现代码如下所示。

```
package com.rss_reader.data;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Vector;

public class RSSFeed
{
	private String title = null;
	private String pubdate = null;
	private int itemcount = 0;
	private List<RSSItem> itemlist;


	 public RSSFeed()
	{
		itemlist = new Vector(0);
	}
	public int addItem(RSSItem item)
```

```
    {
        itemlist.add(item);
        itemcount++;
        return itemcount;
    }
    public RSSItem getItem(int location)
    {
        return itemlist.get(location);
    }
    public List getAllItems()
    {
        return itemlist;
    }
    public List getAllItemsForListView(){
        List<Map<String, Object>> data = new ArrayList<Map<String, Object>>();
        int size = itemlist.size();
        for(int i=0;i<size;i++){
            HashMap<String, Object> item = new HashMap<String, Object>();
            item.put(RSSItem.TITLE, itemlist.get(i).getTitle());
            item.put(RSSItem.PUBDATE, itemlist.get(i).getPubDate());
            data.add(item);
        }
        return data;
    }
    int getItemCount()
    {
        return itemcount;
    }
    public void setTitle(String title)
    {
        this.title = title;
    }
    public void setPubDate(String pubdate)
    {
        this.pubdate = pubdate;
    }
    public String getTitle()
    {
        return title;
    }
    public String getPubDate()
    {
        return pubdate;
    }



}
```

2. RSSItem 类

RSSFeed 类用于和一个 RSS 中的 Item 标签相对应，它里面的属性和 item 里面的属性一样。RSSItem 类的具体实现代码如下所示。

```
package com.rss_reader.data;

public class RSSItem
{
	public static final String TITLE="title";
	public static final String PUBDATE="pubdate";
	private String title = null;
	private String description = null;
	private String link = null;
	private String category = null;
	private String pubdate = null;


	public RSSItem()
	{
	}
	public	void setTitle(String title)
	{
		this.title = title;
	}
	public	void setDescription(String description)
	{
		this.description = description;
	}
	public	void setLink(String link)
	{
		this.link = link;
	}
	public	void setCategory(String category)
	{
		this.category = category;
	}
	public	void setPubDate(String pubdate)
	{
		this.pubdate = pubdate;
	}
	public String getTitle()
	{
		return title;
	}
	public	String getDescription()
	{
		return description;
	}
	public	String getLink()
	{
```

```
            return link;
        }
        public String getCategory()
        {
            return category;
        }
        public String getPubDate()
        {
            return pubdate;
        }
        public String toString()
        {
            if (title.length() > 20)
            {
                return title.substring(0, 42) + "...";
            }
            return title;
        }
}
```

17.2.2 主程序文件 ActivityMain.java

主程序文件 ActivityMain.java 是本项目的入口，在此 Activity 中得到了服务器端的 RSSFeed，经过解析后将里面的内容以 ListView 的形式显示出来。下面开始讲解其具体实现流程。

（1）先引入相关 class 类，然后设置目标 RSS 的源地址为 http://feed.feedsky.com/woshiyigebing12345，最后通过 showListView()方法将获取的 RSS 信息以列表形式显示出来。具体代码如下所示。

```
package com.rss_reader;

import java.net.URL;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.AdapterView.OnItemClickListener;

import com.rss_reader.data.RSSFeed;
import com.rss_reader.data.RSSItem;
import com.rss_reader.sax.RSSHandler;
```

```
public class ActivityMain extends Activity implements OnItemClickListener
{
//      public final String RSS_URL = "http://rubyjin.cn/blog/rss";

            public final String RSS_URL = " http://feed.feedsky.com/woshiyigebing12345";

      public final String tag = "RSSReader";
      private RSSFeed feed = null;

      /** Called when the activity is first created. */

     public void onCreate(Bundle icicle) {
          super.onCreate(icicle);
          setContentView(R.layout.main);
          feed = getFeed(RSS_URL);
          showListView();

     }
```

（2）定义方法 getFeed(String urlString)，用于得到一个 RSSFeed，即从服务器端请求 RSS feed，并进行了解析，将解析后的内容都放在 RSSFeed 的一个实例里。上述解析过程是通过 SAX 实现的，具体流程如下。

- 第一步：新建工厂类SAXParserFactory。
- 第二步：工厂类产出一个SAX解析类SAXParser。
- 第三步：从SAXParser中得到一个XMLReader实例，XMLReader是一个接口，此接口中定义了一些解析XML的回调函数。
- 第四步：把编写的Handler注册到XMLReader中去。
- 第五步：将一个XML文档或资源变成一个Java可以处理的InputStream流后，解析工作开始。

方法 getFeed(String urlString)的具体代码如下所示。

```
private RSSFeed getFeed(String urlString)
{
      try
      {
          URL url = new URL(urlString);
        /*新建工厂类 SAXParserFactory*/
         SAXParserFactory factory = SAXParserFactory.newInstance();
        /*工厂类产出一个 SAX 解析类 SAXParser */
         SAXParser parser = factory.newSAXParser();
        /*从 SAXParser 中得到一个 XMLReader 实例*/
         XMLReader xmlreader = parser.getXMLReader();
         /*把编写的 Handler 注册到 XMLReader 中  */
         RSSHandler rssHandler = new RSSHandler();
         xmlreader.setContentHandler(rssHandler);

         /*将一个 XML 文档或资源变成一个 Java 可以处理的 InputStream 流后，解析工作开始*/
         InputSource is = new InputSource(url.openStream());
```

```
            xmlreader.parse(is);

            return rssHandler.getFeed();
        }
        catch (Exception ee)
        {

            return null;
        }
}
```

（3）定义方法 showListView()来列表显示获取的 RSS，这样 ListView 和一个 SimpleAdapter 实现了绑定。具体代码如下所示。

```
private void showListView()
{
    ListView itemlist = (ListView) findViewById(R.id.itemlist);
    if (feed == null)
    {
     setTitle("访问的 RSS 无效");
     return;
    }
    SimpleAdapter adapter = new SimpleAdapter(this, feed.getAllItemsForListView(),
            android.R.layout.simple_list_item_2, new String[] { RSSItem.TITLE,RSSItem.PUBDATE },
            new int[] { android.R.id.text1 , android.R.id.text2});
    itemlist.setAdapter(adapter);
    itemlist.setOnItemClickListener(this);
    itemlist.setSelection(0);

}
```

（4）定义方法 onItemClick()，用于处理列表的单击事件，当单击后会显示此 RSS 信息的链接地址，用户单击后可以通过浏览器来到目标地址。具体代码如下所示。

```
public void onItemClick(AdapterView parent, View v, int position, long id)
{
        Intent itemintent = new Intent(this,ActivityShowDescription.class);

        Bundle b = new Bundle();
        b.putString("title", feed.getItem(position).getTitle());
        b.putString("description", feed.getItem(position).getDescription());
        b.putString("link", feed.getItem(position).getLink());
        b.putString("pubdate", feed.getItem(position).getPubDate());

        itemintent.putExtra("android.intent.extra.rssItem", b);
       startActivityForResult(itemintent, 0);
}
```

17.2.3 实现 ContentHandler

ContentHandler 是一个特殊的 SAX 接口，位于 org.xml.sax.ContentHandler。在我们解析 XML 时，大多数步骤都是固定不变的，但是关于 ContentHandler 的实现却是不同的。实现 ContentHandler 是解析

XML 中最重要、最关键的步骤之一，下面将开始讲解其具体实现流程。

（1）声明 RSSHandler 类，声明继承于 DefaultHandler 的类。DefaultHandler 类是一个基类，此类里简单实现了一个 ContentHandler，只需重写里面的重要方法即可。具体代码如下所示。

```
package com.rss_reader.sax;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

import android.util.Log;

import com.rss_reader.data.RSSFeed;
import com.rss_reader.data.RSSItem;

public class RSSHandler extends DefaultHandler
{

	RSSFeed rssFeed;
	RSSItem rssItem;
	String lastElementName = "";
	final int RSS_TITLE = 1;
	final int RSS_LINK = 2;
	final int RSS_DESCRIPTION = 3;
	final int RSS_CATEGORY = 4;
	final int RSS_PUBDATE = 5;

	int currentstate = 0;

	public RSSHandler()
	{
	}

	public RSSFeed getFeed()
	{
		return rssFeed;
	}
```

（2）分别重写 startDocument()和 endDocument()，通常将正式解析前的初始化工作放到 startDocument()中，将一些收尾性工作放到 endDocument()中。具体代码如下所示。

```
public void startDocument() throws SAXException
{
	rssFeed = new RSSFeed();
	rssItem = new RSSItem();

}
public void endDocument() throws SAXException
{
}
```

（3）重写 startElement，当 XML 解析器遇到 XML 文档流里的 tag 时，将会调用此函数。在此函数内部通常是通过参数 localName 进行判断并进行一些操作处理的。具体代码如下所示。

```
    public void startElement(String namespaceURI, String localName,String qName, Attributes atts) throws SAXException
    {
        if (localName.equals("channel"))
        {
            currentstate = 0;
            return;
        }
        if (localName.equals("item"))
        {
            rssItem = new RSSItem();
            return;
        }
        if (localName.equals("title"))
        {
            currentstate = RSS_TITLE;
            return;
        }
        if (localName.equals("description"))
        {
            currentstate = RSS_DESCRIPTION;
            return;
        }
        if (localName.equals("link"))
        {
            currentstate = RSS_LINK;
            return;
        }
        if (localName.equals("category"))
        {
            currentstate = RSS_CATEGORY;
            return;
        }
        if (localName.equals("pubDate"))
        {
            currentstate = RSS_PUBDATE;
            return;
        }

        currentstate = 0;
    }
```

（4）重写 endElement，此方法和 startElement 方法相对应，当解析 tag 完毕后执行此方法。如果解析一个 item 节点结束，就将 RSSItem 添加到 RSSFeed 中。具体代码如下所示。

```
public void endElement(String namespaceURI, String localName, String qName) throws SAXException
{

    //如果解析一个 item 节点结束，就将 RSSItem 添加到 RSSFeed 中
    if (localName.equals("item"))
    {
        rssFeed.addItem(rssItem);
```

```
            return;

        }
}
```

（5）重写 characters()方法，此方法是一个回调方法，当解析完 startElement()方法后，解析完节点内容后会执行此方法，并且参数 ch[ ]就是节点的内容。具体代码如下所示。

```
public void characters(char ch[ ], int start, int length)
{
    String theString = new String(ch,start,length);

    switch (currentstate)
    {
        case RSS_TITLE:
            rssItem.setTitle(theString);
            currentstate = 0;
            break;
        case RSS_LINK:
            rssItem.setLink(theString);
            currentstate = 0;
            break;
        case RSS_DESCRIPTION:
            rssItem.setDescription(theString);
            currentstate = 0;
            break;
        case RSS_CATEGORY:
            rssItem.setCategory(theString);
            currentstate = 0;
            break;
        case RSS_PUBDATE:
            rssItem.setPubDate(theString);
            currentstate = 0;
            break;
        default:
            return;
     }

    }
}
```

17.2.4　主程序文件 ActivityShowDescription.java

主程序文件 ActivityShowDescription.java 的功能是显示某列表信息的详细信息。当单击列表中的某一项后会进入到此界面。如果程序出错，则 content 显示出错提示；如运行正确则在 content 中分别显示 title、pubdate 和 description。具体代码如下所示。

```
package com.rss_reader;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
```

```
import android.widget.TextView;
import android.content.Intent;
import android.view.*;

public class ActivityShowDescription extends Activity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.showdescription);
        String content = null;
        Intent startingIntent = getIntent();

        if (startingIntent != null) {
            Bundle bundle = startingIntent
                    .getBundleExtra("android.intent.extra.rssItem");
            if (bundle == null) {
                content = "不好意思程序出错啦";
            } else {
                content = bundle.getString("title") + "\n\n"
                        + bundle.getString("pubdate") + "\n\n"
                        + bundle.getString("description").replace('\n', ' ')
                        + "\n\n 详细信息请访问以下网址:\n" + bundle.getString("link");
            }
        } else {
            content = "不好意思程序出错啦";

        }

        TextView textView = (TextView) findViewById(R.id.content);
        textView.setText(content);

        Button backbutton = (Button) findViewById(R.id.back);

        backbutton.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                finish();
            }
        });
    }
}
```

17.2.5 主布局文件 main.xml

主布局文件 main.xml 用于定义系统初始主界面，即列表显示获取的 RSS 信息。具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<ListView
```

```
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/itemlist"

        />
</LinearLayout>
```

17.2.6 详情主布局文件 showdescription.xml

当用户单击列表信息后，会进入信息详情界面，此界面是由布局文件 showdescription.xml 定义的。具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
<TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:autoLink="all"
     android:text=""
     android:id="@+id/content"
     android:layout_weight="1.0"
     />
<Button
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="返回"
      android:id="@+id/back"
      />

</LinearLayout>
```

至此，整个实例介绍完毕。运行后将获取指定 RSS 中的信息，如图 17-2 所示。单击某条信息后会显示此信息的相关描述性信息，如图 17-3 所示。



图 17-2 初始效果



图 17-3 详情界面

单击图 17-3 中的链接后，能够显示此条 RSS 的详细信息，如图 17-4 所示。

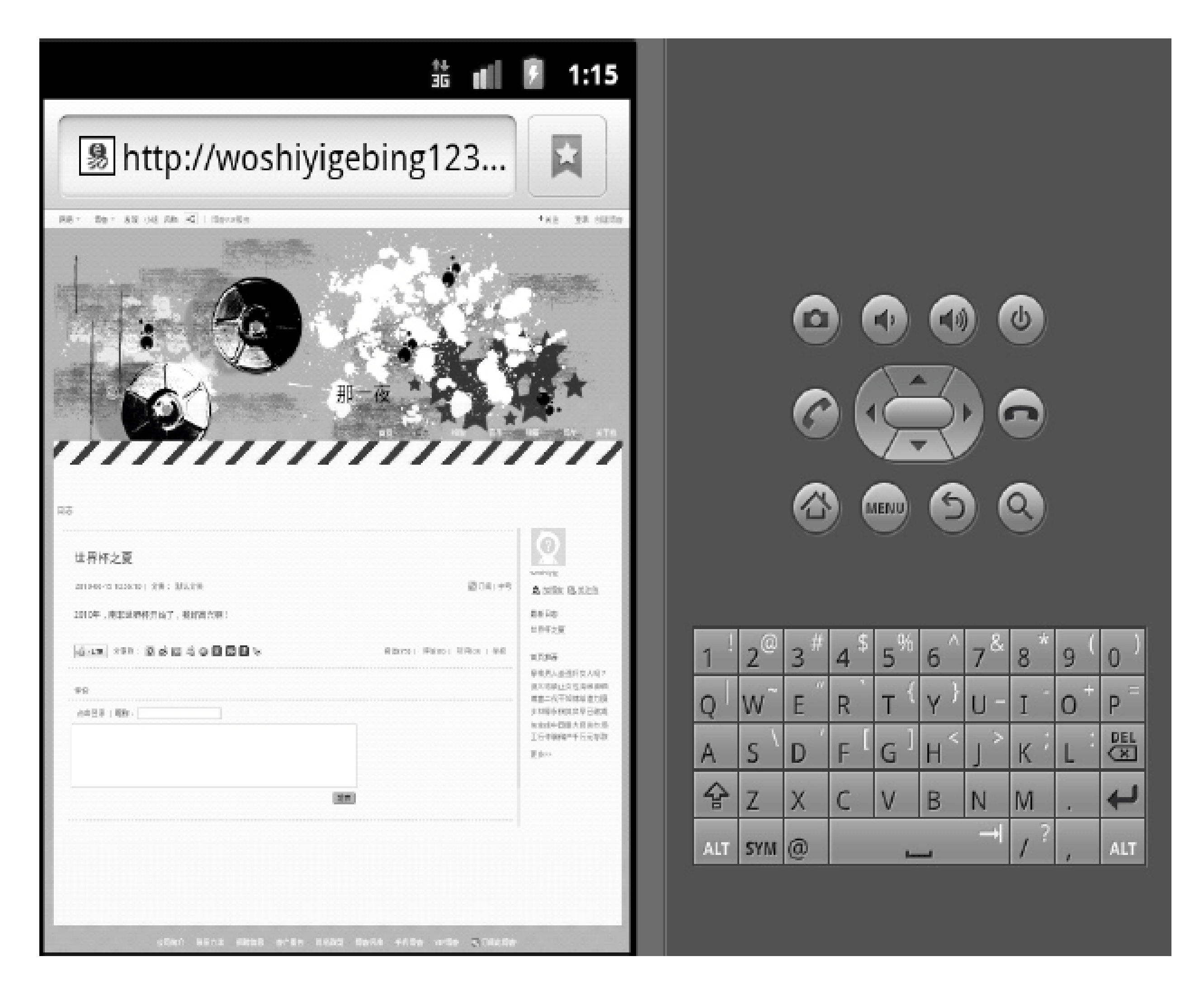


图 17-4　详细信息

本实例默认显示的是博客 http://woshiyigebing12345.blog.163.com/中的信息。读者也可以指定显示其他 RSS 信息，在使用时可以登录 http://www.feedsky.com/来设置不同的 RSS 订阅。具体设置流程如下。

（1）来到 http://www.feedsky.com/主界面，如图 17-5 所示。



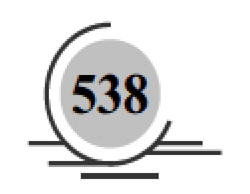
图 17-5　feedsky.com 主界面

（2）在图 17-5 顶部的文本框中输入要显示信息的博客地址、Feed 地址或 QQ 号码，然后单击“下一步”按钮，如图 17-6 所示。



图 17-6　输入设置的博客、QQ 或 Feed 地址

（3）在弹出界面中分别输入名称、描述和 Tag 的信息，并设定永久性 Feed 地址，如图 17-7 所示。



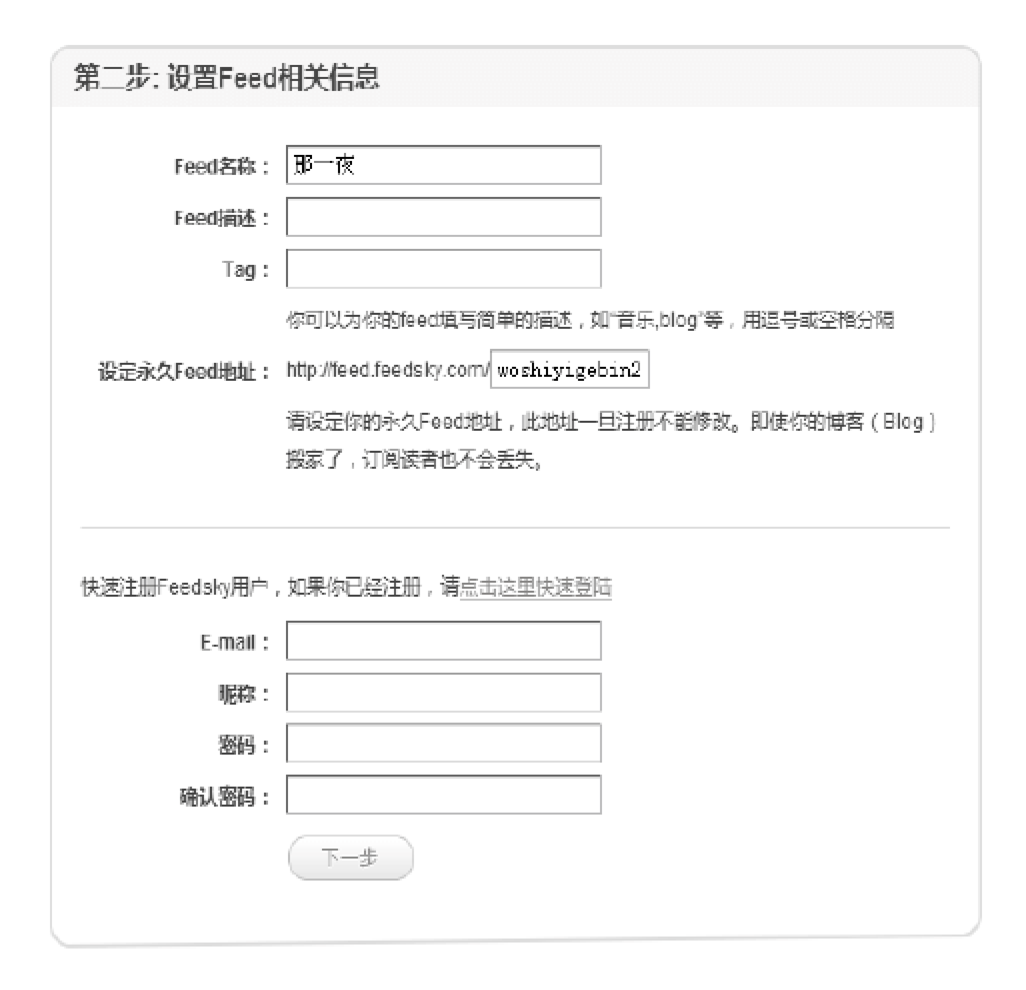


图 17-7　添加 Feed 界面

图 17-7 中的永久 Feed 地址就是 RSS 的源地址，这样就可以将此地址添加到实例中，从而显示此地址的 RSS 资源信息，也就是显示博客 http://woshiyigebing12345.blog.163.com/中的信息。

17.3　打包、签名和发布

知识点讲解：光盘:视频\知识点\第 17 章\打包、签名和发布.avi

当一个 Android 项目开发完毕后，需要打包和签名处理，这样才能放到手机中使用，当然也可以发布到 Market 上去赚钱。下面开始讲解打包、签名、发布 Android 程序的具体过程。

17.3.1　申请会员

即去 Market 申请成为会员，具体流程如下：

（1）登录 http://market.android/publish/signup，如图 17-8 所示。

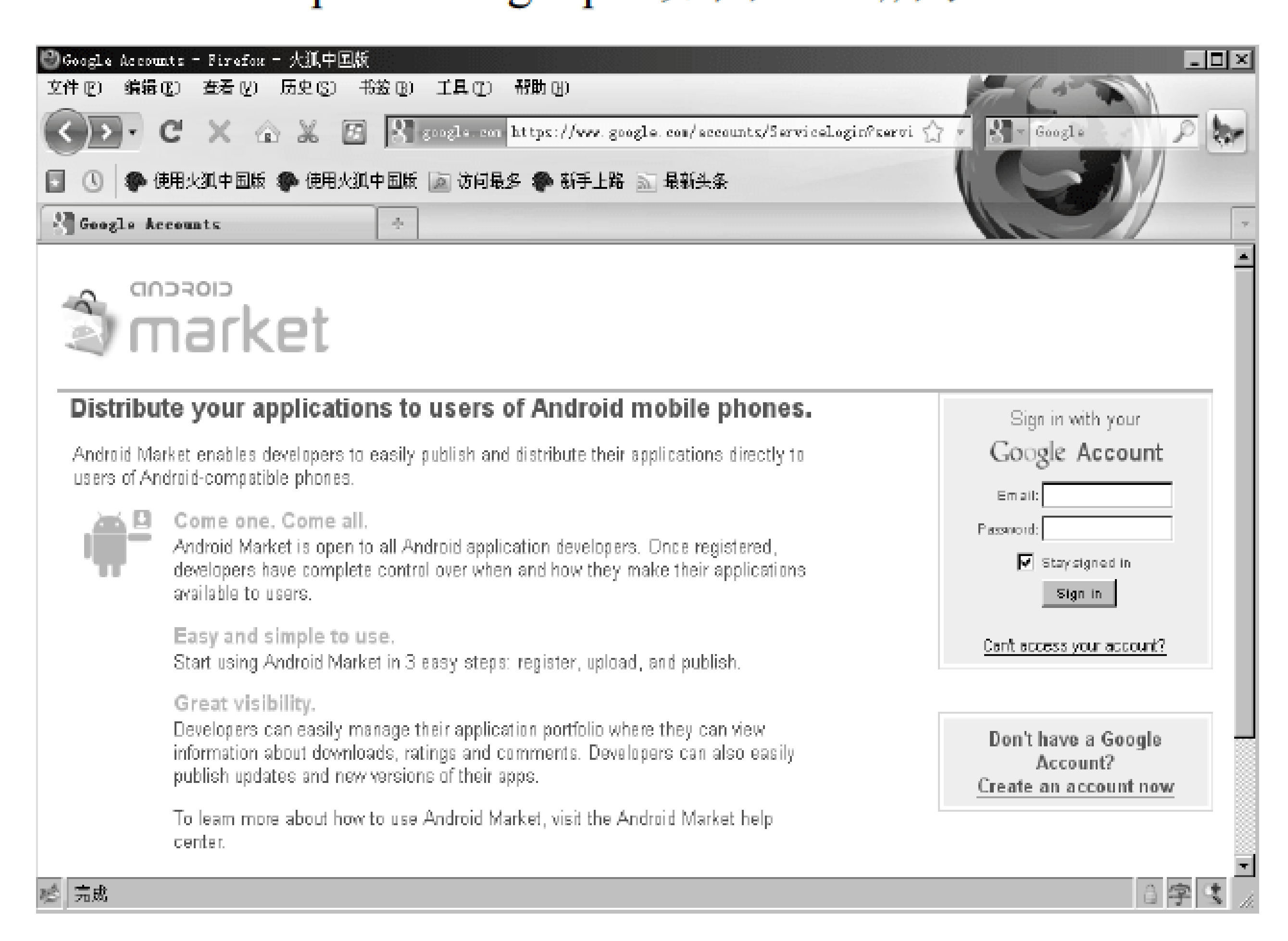


图 17-8　登录 Market

（2）单击链接 Create an account now，来到注册页面，如图 17-9 所示。

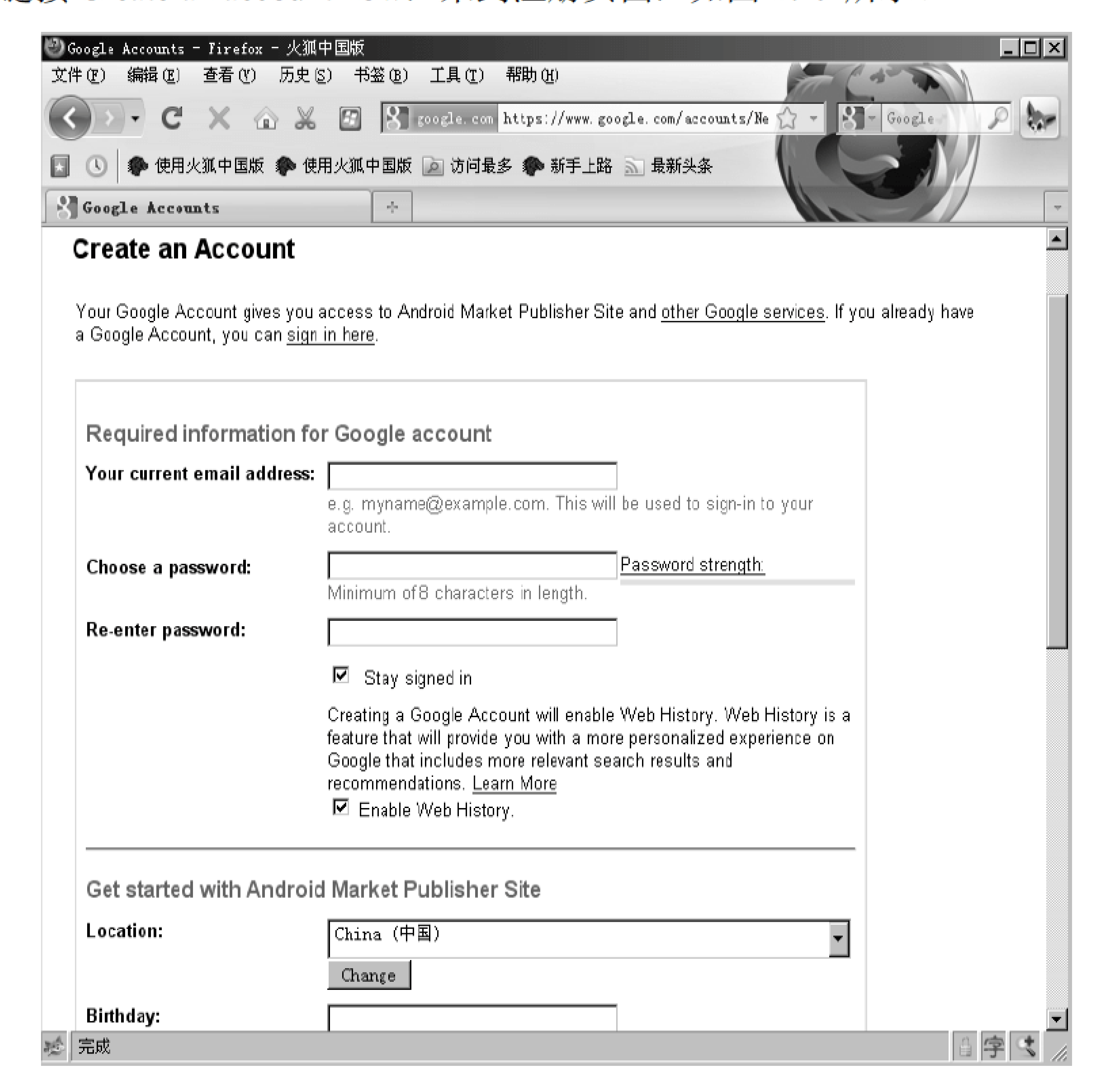


图 17-9　注册界面

（3）单击同意协议后来到下一步页面，在此输入手机号码，如图 17-10 所示。

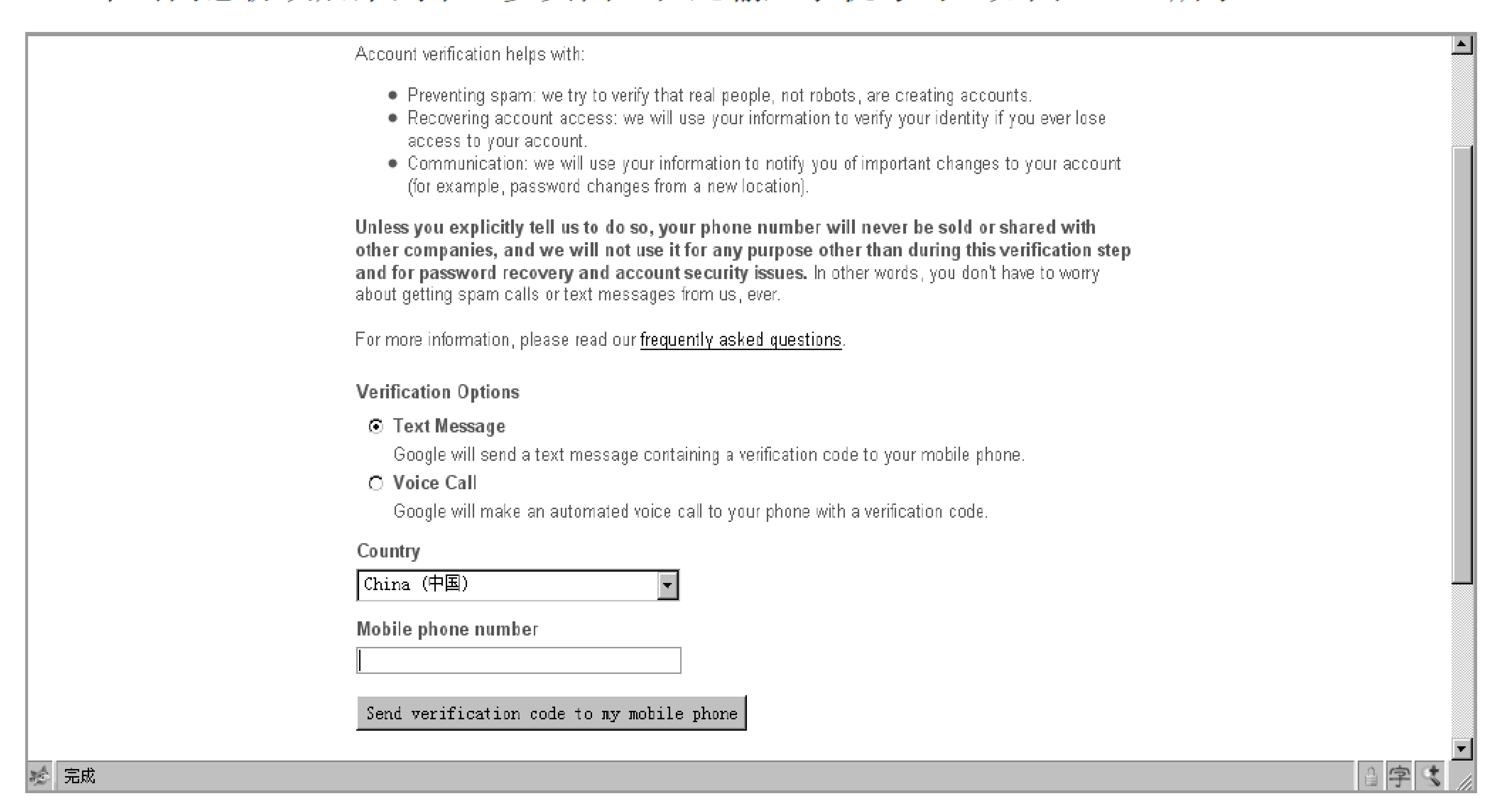


图 17-10　输入手机号码

（4）在新界面中输入手机获取的验证码，如图 17-11 所示。

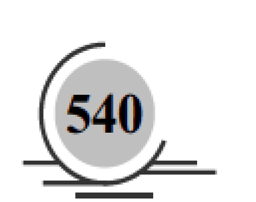




图 17-11　输入验证码

（5）验证通过后，在新界面中继续输入信息，如图 17-12 所示。



图 17-12　输入信息

（6）单击 Continue 按钮后，提示需要花费 25 美元，支付后才能成为正式会员，如图 17-13 所示。



图 17-13　需要支付界面

（7）单击 Google Checkout 按钮来到支付界面，如图 17-14 所示。

图 17-14 支付界面

在此输入你的信用卡信息，完成支付后即可成为正式会员。

17.3.2 生成签名文件

Android 程序的签名和 Symbian 类似都可以自签名（Self-signed），但是在 Android 平台中证书初期还显得形同虚设，平时开发时通过 ADB 接口上传的程序会自动被签有 Debug 权限的程序。需要签名验证，在上传程序到 Android Market 上时大家都已经发现这个问题了。

Android 签名文件的制作方法有以下两种。

1. 第一种：命令行生成

具体流程如下。

（1）cmd 命令如下

```
keytool -genkey -alias android123.keystore -keyalg RSA -validity 20000 -keystore android123.keystore
然后一次提示用户输入如下信息：
输入 keystore 密码：[密码不回显]
再次输入新密码：[密码不回显]
您的名字与姓氏是什么？
[Unknown]：  android123
您的组织单位名称是什么？
[Unknown]：  www.android123.com.cn
您的组织名称是什么？
[Unknown]：  www.android123.com.cn
您的组织名称是什么？
[Unknown]：  www.android123.com.cn
您所在的城市或区域名称是什么？
[Unknown]：  New York
您所在的州或省份名称是什么？
[Unknown]：  New York
该单位的两字母国家代码是什么
[Unknown]：  CN
```

```
CN=android123, OU=www.android123.com.cn, O=www.android123.com.cn, L=New York, ST
=New York, C=CN 正确吗?
[否]: Y
输入<android123.keystore>的主密码（如果和 keystore 密码相同，按回车）:
```

其中参数-validity 为证书有效天数，这里我们写的大于 200 天。另外，在输入密码时没有回显，直接输入就可以了，一般位数建议使用 20 位，最后需要记下来后面还要用。接下来就可以为 apk 文件签名了。

（2）执行

```
jarsigner -verbose -keystore android123.keystore -signedjar android123_signed.apk android123.apk
android123.keystore
```

这样就可以生成签名的 apk 文件，假设输入文件 android123.apk，则最终生成 android123_signed.apk 为 Android 签名后的 APK 执行文件。

注意： keytool用法和jarsigner用法总结

（1）keytool 用法

```
-certreq [-v] [-protected]
              [-alias <别名>] [-sigalg <sigalg>]
              [-file <csr_file>] [-keypass <密钥库口令>]
              [-keystore <密钥库>] [-storepass <存储库口令>]
              [-storetype <存储类型>] [-providername <名称>]
              [-providerclass <提供方类名称> [-providerarg <参数>]] ...
              [-providerpath <路径列表>]

-changealias [-v] [-protected] -alias <别名> -destalias <目标别名>
              [-keypass <密钥库口令>]
              [-keystore <密钥库>] [-storepass <存储库口令>]
              [-storetype <存储类型>] [-providername <名称>]
              [-providerclass <提供方类名称> [-providerarg <参数>]] ...
              [-providerpath <路径列表>]

-delete [-v] [-protected] -alias <别名>
              [-keystore <密钥库>] [-storepass <存储库口令>]
              [-storetype <存储类型>] [-providername <名称>]
              [-providerclass <提供方类名称> [-providerarg <参数>]] ...
              [-providerpath <路径列表>]

-exportcert [-v] [-rfc] [-protected]
              [-alias <别名>] [-file <认证文件>]
              [-keystore <密钥库>] [-storepass <存储库口令>]
              [-storetype <存储类型>] [-providername <名称>]
```

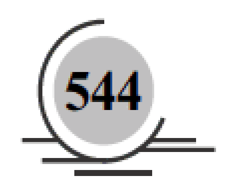
```
            [-providerclass <提供方类名称> [-providerarg <参数>]] ...
            [-providerpath <路径列表>]

-genkeypair [-v] [-protected]
            [-alias <别名>]
            [-keyalg <keyalg>] [-keysize <密钥大小>]
            [-sigalg <sigalg>] [-dname <dname>]
            [-validity <valDays>] [-keypass <密钥库口令>]
            [-keystore <密钥库>] [-storepass <存储库口令>]
            [-storetype <存储类型>] [-providername <名称>]
            [-providerclass <提供方类名称> [-providerarg <参数>]] ...
            [-providerpath <路径列表>]

-genseckey [-v] [-protected]
            [-alias <别名>] [-keypass <密钥库口令>]
            [-keyalg <keyalg>] [-keysize <密钥大小>]
            [-keystore <密钥库>] [-storepass <存储库口令>]
            [-storetype <存储类型>] [-providername <名称>]
            [-providerclass <提供方类名称> [-providerarg <参数>]] ...
            [-providerpath <路径列表>]

-help
-importcert [-v] [-noprompt] [-trustcacerts] [-protected]
            [-alias <别名>]
            [-file <认证文件>] [-keypass <密钥库口令>]
            [-keystore <密钥库>] [-storepass <存储库口令>]
            [-storetype <存储类型>] [-providername <名称>]
            [-providerclass <提供方类名称> [-providerarg <参数>]] ...
            [-providerpath <路径列表>]

-importkeystore [-v]
            [-srckeystore <源密钥库>] [-destkeystore <目标密钥库>]
            [-srcstoretype <源存储类型>] [-deststoretype <目标存储类型>]
            [-srcstorepass <源存储库口令>] [-deststorepass <目标存储库口令>]
            [-srcprotected] [-destprotected]
            [-srcprovidername <源提供方名称>]
            [-destprovidername <目标提供方名称>]
            [-srcalias <源别名> [-destalias <目标别名>]
              [-srckeypass <源密钥库口令>] [-destkeypass <目标密钥库口令>]]
```



[-noprompt]
[-providerclass <提供方类名称> [-providerarg <参数>]] ...
[-providerpath <路径列表>]

-keypasswd [-v] [-alias <别名>]
[-keypass <旧密钥库口令>] [-new <新密钥库口令>]
[-keystore <密钥库>] [-storepass <存储库口令>]
[-storetype <存储类型>] [-providername <名称>]
[-providerclass <提供方类名称> [-providerarg <参数>]] ...
[-providerpath <路径列表>]

-list [-v | -rfc] [-protected]
[-alias <别名>]
[-keystore <密钥库>] [-storepass <存储库口令>]
[-storetype <存储类型>] [-providername <名称>]
[-providerclass <提供方类名称> [-providerarg <参数>]] ...
[-providerpath <路径列表>]

-printcert [-v] [-file <认证文件>]

-storepasswd [-v] [-new <新存储库口令>]
[-keystore <密钥库>] [-storepass <存储库口令>]
[-storetype <存储类型>] [-providername <名称>]
[-providerclass <提供方类名称> [-providerarg <参数>]] ...
[-providerpath <路径列表>]

（2）jarsigner 用法
[选项] jar 文件别名
jarsigner -verify [选项] jar 文件

[-keystore <url>]	密钥库位置
[-storepass <口令>]	用于密钥库完整性的口令
[-storetype <类型>]	密钥库类型
[-keypass <口令>]	专用密钥的口令（如果不同）
[-sigfile <文件>]	.SF/.DSA 文件的名称
[-signedjar <文件>]	已签名的 JAR 文件的名称
[-digestalg <算法>]	摘要算法的名称
[-sigalg <算法>]	签名算法的名称
[-verify]	验证已签名的 JAR 文件
[-verbose]	签名/验证时输出详细信息

[-certs]	输出详细信息和验证时显示证书
[-tsa <url>]	时间戳机构的位置
[-tsacert <别名>]	时间戳机构的公共密钥证书
[-altsigner <类>]	替代的签名机制的类名
[-altsignerpath <路径列表>]	替代的签名机制的位置
[-internalsf]	在签名块内包含 .SF 文件
[-sectionsonly]	不计算整个清单的散列
[-protected]	密钥库已保护验证路径
[-providerName <名称>]	提供者名称
[-providerClass <类>	加密服务提供者的名称
[-providerArg <参数>]] ...	主类文件和构造函数参数

2. 第二种：Eclipse 的 ADT 生成

实际上，使用 Eclipse 可以更加直观、方便地生成签名文件，具体流程如下。

（1）右键单击 Eclipse 项目名，在弹出快捷菜单中依次选择 Android Tools | Export Signed Application Package 命令，如图 17-15 所示。



图 17-15　选择导出

（2）在弹出的对话框中选择要导出的项目，如图 17-16 所示。

（3）单击 Next 按钮，在弹出的对话框中选中 Create new keystore 单选按钮，然后分别输入文件名和密码，如图 17-17 所示。



图 17-16　选择要导出的项目

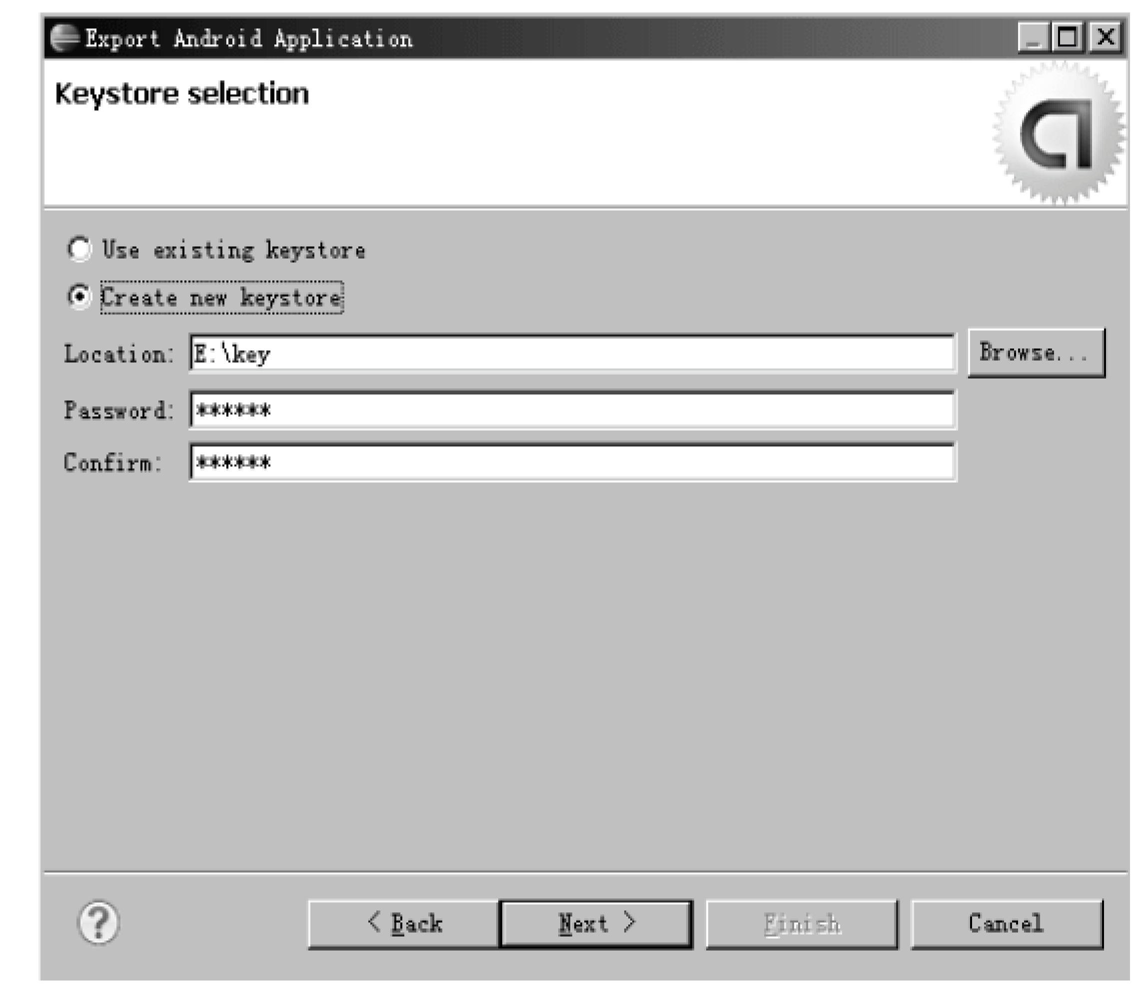
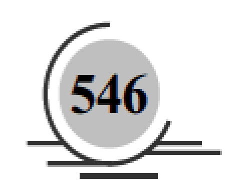


图 17-17　文件名和密码

（4）单击 Next 按钮，在弹出的界面中输入签名文件路径，如图 17-18 所示。

（5）单击 Next 按钮，在弹出的界面中依次输入签名文件的相关信息，如图 17-19 所示。



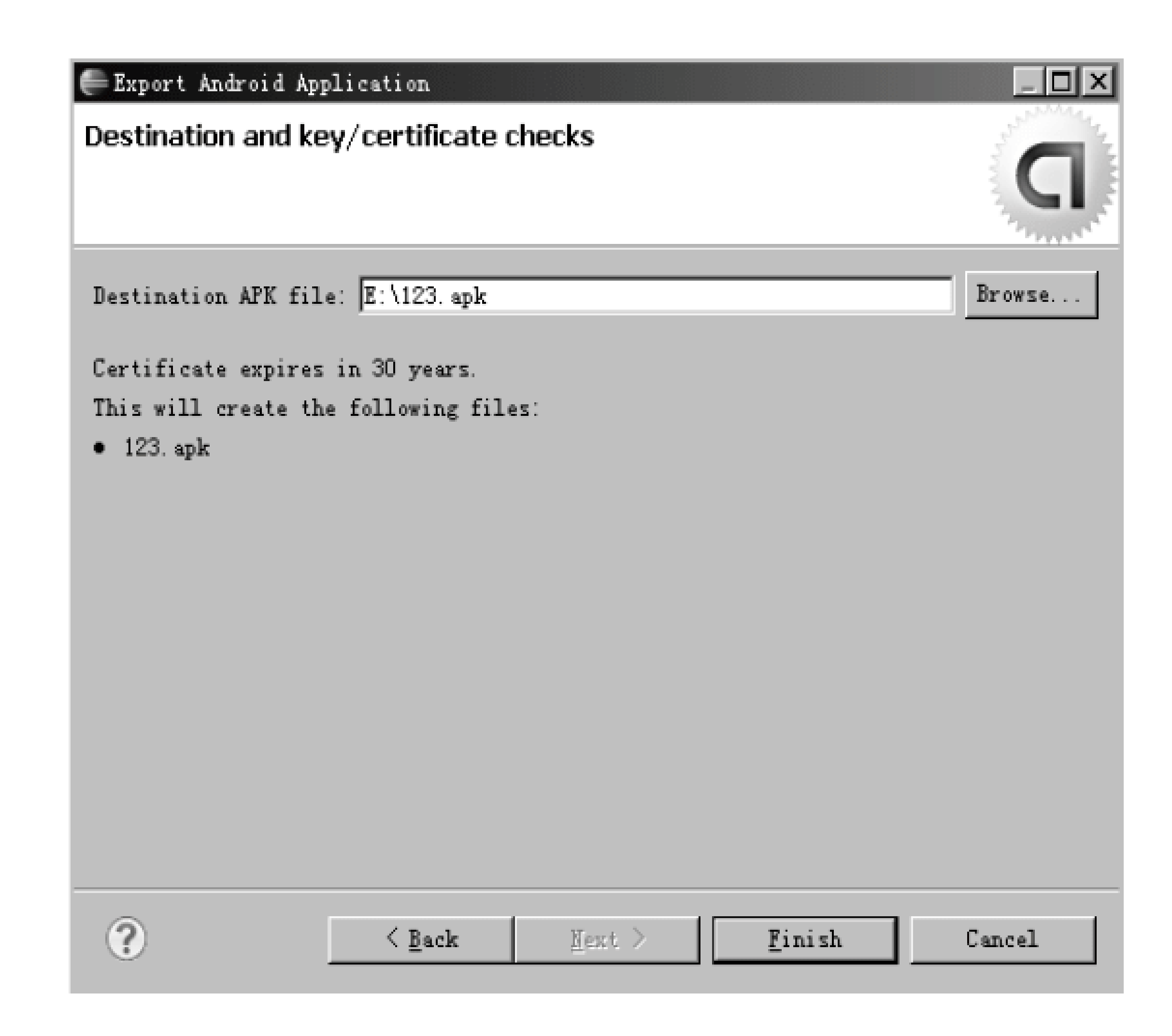


图 17-18　输入信息

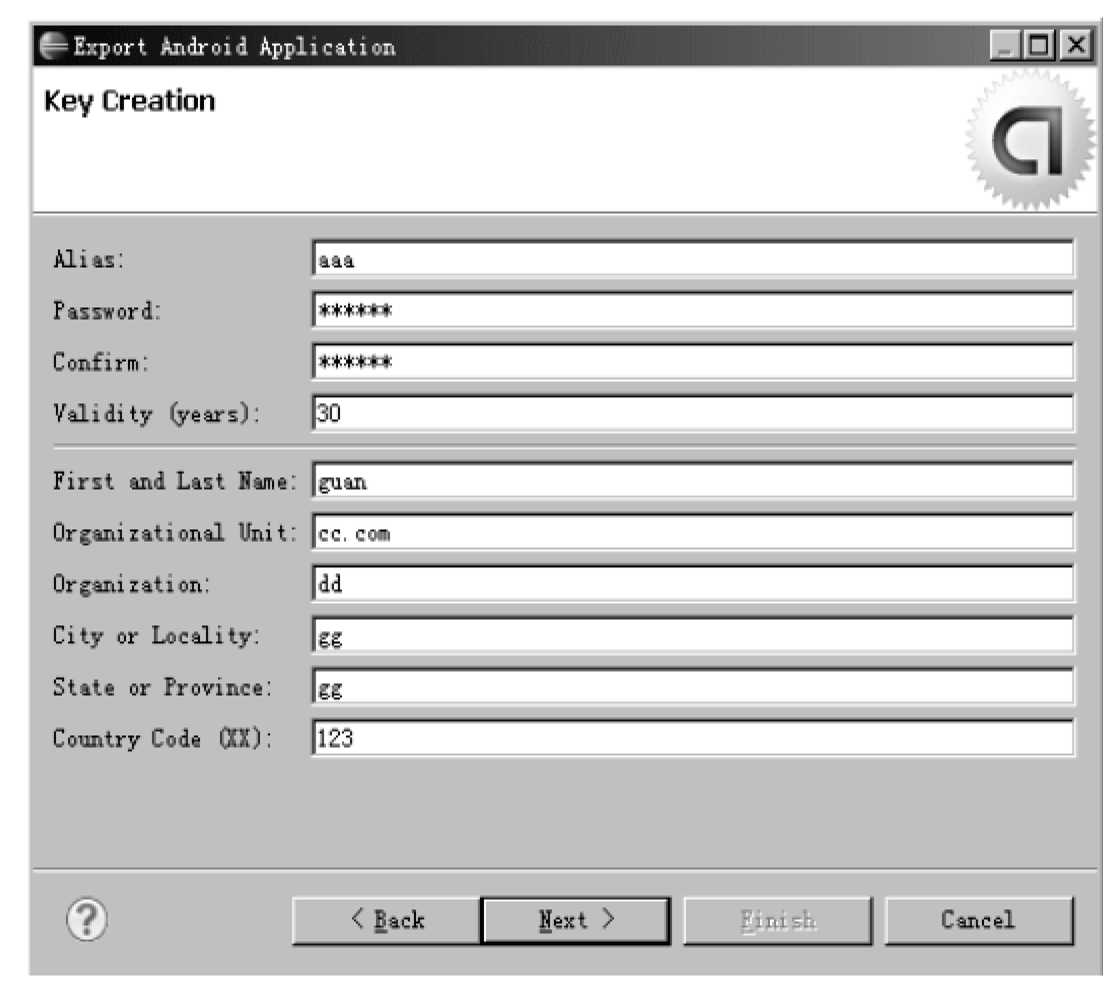


图 17-19　输入信息

（6）单击 Next 按钮后完成签名文件的创建。

17.3.3　使用签名文件

生成签名文件后，就可以使用它了，在此也有两种方式。

1. 第一种：命令行

（1）假设生成的签名文件是 ChangeBackgroundWidget.apk，则最终生成 ChangeBackground-Widget_signed.apk 为 Android 签名后的 APK 执行文件。

输入以下命令行：

```
jarsigner -verbose -keystore ChangeBackgroundWidget.keystore -signedjar ChangeBackgroundWidget_signed.
apk ChangeBackgroundWidget.apk ChangeBackgroundWidget.keystore
```

上面命令中间不换行。

（2）按 Enter 键，根据提示输入密钥库的口令短语（即密码），详细信息如下。

输入密钥库的口令短语：

```
正在添加：META-INF/MANIFEST.MF
正在添加：META-INF/CHANGEBA.SF
正在添加：META-INF/CHANGEBA.RSA
正在签名：res/drawable/icon.png
正在签名：res/drawable/icon_audio.png
正在签名：res/drawable/icon_exit.png
正在签名：res/drawable/icon_folder.png
正在签名：res/drawable/icon_home.png
正在签名：res/drawable/icon_img.png
正在签名：res/drawable/icon_left.png
正在签名：res/drawable/icon_mantou.png
正在签名：res/drawable/icon_other.png
正在签名：res/drawable/icon_pause.png
正在签名：res/drawable/icon_play.png
正在签名：res/drawable/icon_return.png
```

```
正在签名：res/drawable/icon_right.png
正在签名：res/drawable/icon_set.png
正在签名：res/drawable/icon_text.png
正在签名：res/drawable/icon_xin.png
正在签名：res/layout/fileitem.xml
正在签名：res/layout/filelist.xml
正在签名：res/layout/main.xml
正在签名：res/layout/widget.xml
正在签名：res/xml/widget_info.xml
正在签名：AndroidManifest.xml
正在签名：resources.arsc
正在签名：classes.dex
```

通过上述过程处理后，即可将未签名文件 ChangeBackgroundWidget.apk 签名为 ChangeBackground-Widget_signed.apk。

在上述方式中，读者可能会遇到以下问题。

问题 1：jarsigner：无法打开 jar 文件 ChangeBackgroundWidget.apk。

解决方法：将要进行签名的 APK 放到对应的文件下，把要签名的 ChangeBackgroundWidget.apk 放到 JDK 的 bin 文件里。

问题 2：jarsigner 无法对 jar 进行签名：java.util.zip.ZipException: invalid entry compressed size (expected 1598 but got 1622 bytes)。

方法 1：Android 开发网提示这些问题主要是由于资源文件造成的，对于 Android 开发来说应该检查 res 文件夹中的文件，逐个排查。这个问题可以通过升级系统的 JDK 和 JRE 版本来解决。

方法 2：这是因为默认给 apk 做了 debug 签名，所以无法做新的签名，这时就必须右键单击工程，选择 Android Tools | Export Unsigned Application Package 命令。或者从 AndroidManifest.xml 的 Exporting 上选择也是一样的。

然后再基于这个导出的 unsigned apk 做签名，导出的时候最好将其目录选在之前产生 keystore 的那个目录下，这样操作起来就方便了。

2. 第二种：直接导入

第二种方式是在 Eclipse 中直接导入的方式，具体过程和 17.3.2 节中第二种方法的过程类似，不再赘述。

17.3.4 发布

发布的过程比较简单，来到 Market，登录个人中心，上传签名后的文件即可，具体操作流程在 Market 站点上有详细说明，为节省篇幅，在此不做详细介绍。

第 18 章 QR 码采集器

QR 是英文 Quick Response 的缩写，即快速反应。QR 码比普通条码可储存更多资料，扫描时的要求很低，无须像普通条码扫描时需直线对准扫描器才能识别。QR 码可以存储名片信息等大容量的内容，包括 WIFI ACCESS、文档、数字、网址等，因此其应用越来越广泛，领域包括电子商务、签到、防伪等。QR 的形式可以一反往常的黑白色调以及单调的方框，制作出很多有趣生动的 QR 二维码。本章将通过一个综合实例的实现过程，详细讲解在 Android 设备中开发一个 QR 码采集器的方法。

18.1 信 息 采 集

知识点讲解：光盘:视频\知识点\第 18 章\信息采集.avi

本项目实例的功能是，快速采集 QR 二维码信息，并且可以将自己的联系信息、网站、邮箱、推特或某个 Android Market 的应用程序生成对应的 QR 二维码。

实例	功能	源码路径
实例 18-1	开发一个 QR 码生成器	光盘:\codes\18\android-quick-response-code-EX

信息采集的功能通过摄像头扫描拍摄 QR 码信息，在本节的内容中，将详细讲解信息采集模块的具体实现流程。

18.1.1 采集界面的主 Activity

本系统信息采集界面的主 Activity 的实现文件是 CaptureActivity.java，功能是调用布局文件 capture.xml 加载显示 UI 控件，通过自定义的 UI 界面来处理 decodBarcodeFormated 的内容。文件 CaptureActivity.java 的具体实现代码如下所示。

```
public class CaptureActivity extends DecoderActivity {

    private static final String TAG = CaptureActivity.class.getSimpleName();
    private static final Set<ResultMetadataType> DISPLAYABLE_METADATA_TYPES = EnumSet.of(Result
MetadataType.ISSUE_NUMBER, ResultMetadataType.SUGGESTED_PRICE,
            ResultMetadataType.ERROR_CORRECTION_LEVEL,
ResultMetadataType.POSSIBLE_COUNTRY);

    private TextView statusView = null;
    private View resultView = null;
    private boolean inScanMode = false;

    @Override
    public void onCreate(Bundle icicle) {
```

```
        super.onCreate(icicle);
        setContentView(R.layout.capture);
        Log.v(TAG, "onCreate()");

        resultView = findViewById(R.id.result_view);
        statusView = (TextView) findViewById(R.id.status_view);

        inScanMode = false;
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.v(TAG, "onDestroy()");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.v(TAG, "onResume()");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.v(TAG, "onPause()");
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            if (inScanMode)
                finish();
            else
                onResume();
            return true;
        }
        return super.onKeyDown(keyCode, event);
    }

    @Override
    public void handleDecode(Result rawResult, Bitmap barcode) {
        drawResultPoints(barcode, rawResult);

        ResultHandler resultHandler = ResultHandlerFactory.makeResultHandler(this, rawResult);
        handleDecodeInternally(rawResult, resultHandler, barcode);
    }

    protected void showScanner() {
        inScanMode = true;
        resultView.setVisibility(View.GONE);
```

```
            statusView.setText(R.string.msg_default_status);
            statusView.setVisibility(View.VISIBLE);
            viewfinderView.setVisibility(View.VISIBLE);
      }

      protected void showResults() {
            inScanMode = false;
            statusView.setVisibility(View.GONE);
            viewfinderView.setVisibility(View.GONE);
            resultView.setVisibility(View.VISIBLE);
      }

      // Put up our own UI for how to handle the decodBarcodeFormated contents.
      private void handleDecodeInternally(Result rawResult, ResultHandler resultHandler, Bitmap barcode) {
            onPause();
            showResults();

            ImageView barcodeImageView = (ImageView) findViewById(R.id.barcode_image_view);
            if (barcode == null) {
                  barcodeImageView.setImageBitmap(BitmapFactory.decodeResource(getResources(),
R.drawable.icon));
            } else {
                  barcodeImageView.setImageBitmap(barcode);
            }

            TextView formatTextView = (TextView) findViewById(R.id.format_text_view);
            formatTextView.setText(rawResult.getBarcodeFormat().toString());

            TextView typeTextView = (TextView) findViewById(R.id.type_text_view);
            typeTextView.setText(resultHandler.getType().toString());

            DateFormat formatter = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.SHORT);
            String formattedTime = formatter.format(new Date(rawResult.getTimestamp()));
            TextView timeTextView = (TextView) findViewById(R.id.time_text_view);
            timeTextView.setText(formattedTime);

            TextView metaTextView = (TextView) findViewById(R.id.meta_text_view);
            View metaTextViewLabel = findViewById(R.id.meta_text_view_label);
            metaTextView.setVisibility(View.GONE);
            metaTextViewLabel.setVisibility(View.GONE);
            Map<ResultMetadataType, Object> metadata = rawResult.getResultMetadata();
            if (metadata != null) {
                  StringBuilder metadataText = new StringBuilder(20);
                  for (Map.Entry<ResultMetadataType, Object> entry : metadata.entrySet()) {
                        if (DISPLAYABLE_METADATA_TYPES.contains(entry.getKey())) {
                              metadataText.append(entry.getValue()).append('\n');
                        }
                  }
                  if (metadataText.length() > 0) {
                        metadataText.setLength(metadataText.length() - 1);
                        metaTextView.setText(metadataText);
```

```
                    metaTextView.setVisibility(View.VISIBLE);
                    metaTextViewLabel.setVisibility(View.VISIBLE);
                }
            }

            TextView contentsTextView = (TextView) findViewById(R.id.contents_text_view);
            CharSequence displayContents = resultHandler.getDisplayContents();
            contentsTextView.setText(displayContents);
            // Crudely scale betweeen 22 and 32 -- bigger font for shorter text
            int scaledSize = Math.max(22, 32 - displayContents.length() / 4);
            contentsTextView.setTextSize(TypedValue.COMPLEX_UNIT_SP, scaledSize);
        }
}
```

布局文件 capture.xml 的具体实现代码如下所示。

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent">
  <SurfaceView android:id="@+id/preview_view"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent">
  </SurfaceView>
  <com.jwetherell.quick_response_code.ViewfinderView
      android:id="@+id/viewfinder_view"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:background="@color/transparent">
  </com.jwetherell.quick_response_code.ViewfinderView>
  <LinearLayout android:id="@+id/result_view"
                android:orientation="horizontal"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:background="@color/result_view"
                android:gravity="center"
                android:padding="4dip">

    <LinearLayout
      android:orientation="vertical"
      android:layout_width="wrap_content"
      android:layout_height="fill_parent"
      android:gravity="right|center_vertical">

      <ImageView android:id="@+id/barcode_image_view"
                 android:layout_width="160dip"
                 android:layout_height="wrap_content"
                 android:maxWidth="160dip"
                 android:maxHeight="160dip"
                 android:layout_marginBottom="4dip"
                 android:adjustViewBounds="true"
                 android:scaleType="centerInside"
                 android:contentDescription="@string/barcode_image">
```

```
</ImageView>

<LinearLayout
  android:orientation="horizontal"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
  <TextView android:id="@+id/format_text_view_label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/msg_default_format"
            android:textColor="@color/result_minor_text"
            android:textStyle="bold"
            android:textSize="14sp"
            android:paddingRight="4dip">
  </TextView>
  <TextView android:id="@+id/format_text_view"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/result_minor_text"
            android:textSize="14sp">
  </TextView>
</LinearLayout>

<LinearLayout
  android:orientation="horizontal"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
  <TextView android:id="@+id/type_text_view_label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/msg_default_type"
            android:textColor="@color/result_minor_text"
            android:textStyle="bold"
            android:textSize="14sp"
            android:paddingRight="4dip">
  </TextView>
  <TextView android:id="@+id/type_text_view"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/result_minor_text"
            android:textSize="14sp">
  </TextView>
</LinearLayout>

<LinearLayout
  android:orientation="horizontal"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
  <TextView android:id="@+id/time_text_view_label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
                    android:text="@string/msg_default_time"
                    android:textColor="@color/result_minor_text"
                    android:textStyle="bold"
                    android:textSize="14sp"
                    android:paddingRight="4dip">
        </TextView>
        <TextView android:id="@+id/time_text_view"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textColor="@color/result_minor_text"
                    android:textSize="14sp">
        </TextView>
    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/meta_text_view_label"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/msg_default_meta"
                    android:textColor="@color/result_minor_text"
                    android:textStyle="bold"
                    android:textSize="14sp"
                    android:paddingRight="4dip">
        </TextView>
        <TextView android:id="@+id/meta_text_view"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textColor="@color/result_minor_text"
                    android:textSize="14sp">
        </TextView>
    </LinearLayout>

</LinearLayout>

<ScrollView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView android:id="@+id/contents_text_view"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textColor="@color/result_text"
                    android:textColorLink="@color/result_text"
                    android:textSize="22sp"
                    android:paddingLeft="12dip">
```

```
            </TextView>
        </LinearLayout>
    </ScrollView>

  </LinearLayout>

  <TextView android:id="@+id/status_view"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_gravity="bottom|center_horizontal"
              android:background="@color/transparent"
              android:text="@string/msg_default_status"
              android:textColor="@color/status_text"
              android:textSize="14sp">
  </TextView>
</FrameLayout>
```

18.1.2　相机采集

在采集外部信息时，是通过设备中自带的摄像头拍照实现的。其中文件 AutoFocusCallback.java 实现自动对焦回调处理，具体实现代码如下所示。

```
final class AutoFocusCallback implements Camera.AutoFocusCallback {
    private static final String TAG = AutoFocusCallback.class.getSimpleName();
    private static final long AUTOFOCUS_INTERVAL_MS = 1500L;
    private Handler autoFocusHandler;
    private int autoFocusMessage;
    void setHandler(Handler autoFocusHandler, int autoFocusMessage) {
        this.autoFocusHandler = autoFocusHandler;
        this.autoFocusMessage = autoFocusMessage;
    }
    @Override
    public void onAutoFocus(boolean success, Camera camera) {
        if (autoFocusHandler != null) {
            Message message = autoFocusHandler.obtainMessage(autoFocusMessage, success);
            // Simulate continuous autofocus by sending a focus request every
            // AUTOFOCUS_INTERVAL_MS milliseconds
            // Log.d(TAG, "Got auto-focus callback; requesting another");
            autoFocusHandler.sendMessageDelayed(message, AUTOFOCUS_INTERVAL_MS);
            autoFocusHandler = null;
        } else {
            Log.d(TAG, "Got auto-focus callback, but no handler for it");
        }
    }
}
```

文件 CameraConfigurationManager.java 实现了相机配置管理器功能，其中涉及了阅读采集、拍照分析和设置相机参数功能。文件 CameraConfigurationManager.java 的具体实现代码如下所示。

```
public final class CameraConfigurationManager {

    private static final String TAG = "CameraConfiguration";
```

```
    private static final int MIN_PREVIEW_PIXELS = 320 * 240; // small screen
    private static final int MAX_PREVIEW_PIXELS = 800 * 480; // large/HD screen

    private final Context context;
    private Point screenResolution;
    private Point cameraResolution;

    public CameraConfigurationManager(Context context) {
        this.context = context;
    }

    /**
     *瞬间读取采集的值
     */
    void initFromCameraParameters(Camera camera) {
        Camera.Parameters parameters = camera.getParameters();
        WindowManager manager = (WindowManager) context.getSystemService(Context.WINDOW_SERVICE);
        Display display = manager.getDefaultDisplay();
        int width = display.getWidth();
        int height = display.getHeight();
        if (width < height) {
            Log.i(TAG, "Display reports portrait orientation; assuming this is incorrect");
            int temp = width;
            width = height;
            height = temp;
        }
        screenResolution = new Point(width, height);
        Log.i(TAG, "Screen resolution: " + screenResolution);
        cameraResolution = findBestPreviewSizeValue(parameters, screenResolution, false);
        Log.i(TAG, "Camera resolution: " + cameraResolution);
    }

    void setDesiredCameraParameters(Camera camera) {
        Camera.Parameters parameters = camera.getParameters();

        if (parameters == null) {
            Log.w(TAG, "Device error: no camera parameters are available. Proceeding without configuration.");
            return;
        }

        initializeTorch(parameters);
        String focusMode = findSettableValue(parameters.getSupportedFocusModes(), Camera.Parameters.
FOCUS_MODE_AUTO, Camera.Parameters.FOCUS_MODE_MACRO);
        if (focusMode != null) {
            parameters.setFocusMode(focusMode);
        }

        parameters.setPreviewSize(cameraResolution.x, cameraResolution.y);
        camera.setParameters(parameters);
    }
```

```
    public Point getCameraResolution() {
        return cameraResolution;
    }

    public Point getScreenResolution() {
        return screenResolution;
    }

    void setTorch(Camera camera, boolean newSetting) {
        Camera.Parameters parameters = camera.getParameters();
        doSetTorch(parameters, newSetting);
        camera.setParameters(parameters);
    }

    private static void initializeTorch(Camera.Parameters parameters) {
        doSetTorch(parameters, Preferences.KEY_FRONT_LIGHT);
    }

    private static void doSetTorch(Camera.Parameters parameters, boolean newSetting) {
        String flashMode;
        if (newSetting) {
            flashMode = findSettableValue(parameters.getSupportedFlashModes(), Camera.Parameters.
FLASH_MODE_TORCH, Camera.Parameters.FLASH_MODE_ON);
        } else {
            flashMode = findSettableValue(parameters.getSupportedFlashModes(), Camera.Parameters.
FLASH_MODE_OFF);
        }
        if (flashMode != null) {
            parameters.setFlashMode(flashMode);
        }
    }

    private static Point findBestPreviewSizeValue(Camera.Parameters parameters, Point screenResolution,
boolean portrait) {
        Point bestSize = null;
        int diff = Integer.MAX_VALUE;
        for (Camera.Size supportedPreviewSize : parameters.getSupportedPreviewSizes()) {
            int pixels = supportedPreviewSize.height * supportedPreviewSize.width;
            if (pixels < MIN_PREVIEW_PIXELS || pixels > MAX_PREVIEW_PIXELS) {
                continue;
            }
            int supportedWidth = portrait ? supportedPreviewSize.height : supportedPreviewSize.width;
            int supportedHeight = portrait ? supportedPreviewSize.width : supportedPreviewSize.height;
            int newDiff = Math.abs(screenResolution.x * supportedHeight - supportedWidth * screenResolution.y);
            if (newDiff == 0) {
                bestSize = new Point(supportedWidth, supportedHeight);
                break;
            }
            if (newDiff < diff) {
                bestSize = new Point(supportedWidth, supportedHeight);
                diff = newDiff;
```

```
                }
            }
            if (bestSize == null) {
                Camera.Size defaultSize = parameters.getPreviewSize();
                bestSize = new Point(defaultSize.width, defaultSize.height);
            }
            return bestSize;
        }

        private static String findSettableValue(Collection<String> supportedValues, String... desiredValues) {
            Log.i(TAG, "Supported values: " + supportedValues);
            String result = null;
            if (supportedValues != null) {
                for (String desiredValue : desiredValues) {
                    if (supportedValues.contains(desiredValue)) {
                        result = desiredValue;
                        break;
                    }
                }
            }
            Log.i(TAG, "Settable value: " + result);
            return result;
        }
}
```

18.1.3 实现取景器功能

文件 PreviewCallback.java 的功能是实现当前相机拍照的预览功能，具体实现代码如下所示。

```
final class PreviewCallback implements Camera.PreviewCallback {
        private static final String TAG = PreviewCallback.class.getSimpleName();

        private final CameraConfigurationManager configManager;
        private Handler previewHandler;
        private int previewMessage;
        PreviewCallback(CameraConfigurationManager configManager) {
            this.configManager = configManager;
        }
        void setHandler(Handler previewHandler, int previewMessage) {
            this.previewHandler = previewHandler;
            this.previewMessage = previewMessage;
        }
        @Override
        public void onPreviewFrame(byte[ ] data, Camera camera) {
            Point cameraResolution = configManager.getCameraResolution();
            Handler thePreviewHandler = previewHandler;
            if (thePreviewHandler != null) {
                Message message = thePreviewHandler.obtainMessage(previewMessage, cameraResolution.x,
cameraResolution.y, data);
                message.sendToTarget();
                previewHandler = null;
```

```
            } else {
                Log.d(TAG, "Got preview callback, but no handler for it");
            }
        }
    }
```

编写文件 ViewfinderView.java 生成一个取景器视图，此视图被覆盖在相机预览界面的顶部，这样增加了取景器的矩形和外面部分的透明性，以及激光扫描器动画和结果值的清晰度。文件 ViewfinderView.java 的具体实现代码如下所示。

```
public final class ViewfinderView extends View {

    private static final int[ ] SCANNER_ALPHA = { 0, 64, 128, 192, 255, 192, 128, 64 };
    private static final long ANIMATION_DELAY = 80L;
    private static final int CURRENT_POINT_OPACITY = 0xA0;
    private static final int MAX_RESULT_POINTS = 20;
    private static final int POINT_SIZE = 6;

    private CameraManager cameraManager;
    private final Paint paint;
    private Bitmap resultBitmap;
    private final int maskColor;
    private final int resultColor;
    private final int frameColor;
    private final int laserColor;
    private final int resultPointColor;
    private int scannerAlpha;
    private List<ResultPoint> possibleResultPoints;
    private List<ResultPoint> lastPossibleResultPoints;

    // This constructor is used when the class is built from an XML resource.
    public ViewfinderView(Context context, AttributeSet attrs) {
        super(context, attrs);

        // Initialize these once for performance rather than calling them every
        // time in onDraw().
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        Resources resources = getResources();
        maskColor = resources.getColor(R.color.viewfinder_mask);
        resultColor = resources.getColor(R.color.result_view);
        frameColor = resources.getColor(R.color.viewfinder_frame);
        laserColor = resources.getColor(R.color.viewfinder_laser);
        resultPointColor = resources.getColor(R.color.possible_result_points);
        scannerAlpha = 0;
        possibleResultPoints = new ArrayList<ResultPoint>(5);
        lastPossibleResultPoints = null;
    }

    public void setCameraManager(CameraManager cameraManager) {
        this.cameraManager = cameraManager;
    }
```

```
@Override
public void onDraw(Canvas canvas) {
    Rect frame = cameraManager.getFramingRect();
    if (frame == null) {
        return;
    }
    int width = canvas.getWidth();
    int height = canvas.getHeight();

    // Draw the exterior (i.e. outside the framing rect) darkened
    paint.setColor(resultBitmap != null ? resultColor : maskColor);
    canvas.drawRect(0, 0, width, frame.top, paint);
    canvas.drawRect(0, frame.top, frame.left, frame.bottom + 1, paint);
    canvas.drawRect(frame.right + 1, frame.top, width, frame.bottom + 1, paint);
    canvas.drawRect(0, frame.bottom + 1, width, height, paint);

    if (resultBitmap != null) {
        // Draw the opaque result bitmap over the scanning rectangle
        paint.setAlpha(CURRENT_POINT_OPACITY);
        canvas.drawBitmap(resultBitmap, null, frame, paint);
    } else {

        // Draw a two pixel solid black border inside the framing rect
        paint.setColor(frameColor);
        canvas.drawRect(frame.left, frame.top, frame.right + 1, frame.top + 2, paint);
        canvas.drawRect(frame.left, frame.top + 2, frame.left + 2, frame.bottom - 1, paint);
        canvas.drawRect(frame.right - 1, frame.top, frame.right + 1, frame.bottom - 1, paint);
        canvas.drawRect(frame.left, frame.bottom - 1, frame.right + 1, frame.bottom + 1, paint);

        // Draw a red "laser scanner" line through the middle to show
        // decoding is active
        paint.setColor(laserColor);
        paint.setAlpha(SCANNER_ALPHA[scannerAlpha]);
        scannerAlpha = (scannerAlpha + 1) % SCANNER_ALPHA.length;
        int middle = frame.height() / 2 + frame.top;
        canvas.drawRect(frame.left + 2, middle - 1, frame.right - 1, middle + 2, paint);

        Rect previewFrame = cameraManager.getFramingRectInPreview();
        float scaleX = frame.width() / (float) previewFrame.width();
        float scaleY = frame.height() / (float) previewFrame.height();

        List<ResultPoint> currentPossible = possibleResultPoints;
        List<ResultPoint> currentLast = lastPossibleResultPoints;
        int frameLeft = frame.left;
        int frameTop = frame.top;
        if (currentPossible.isEmpty()) {
            lastPossibleResultPoints = null;
        } else {
            possibleResultPoints = new ArrayList<ResultPoint>(5);
            lastPossibleResultPoints = currentPossible;
            paint.setAlpha(CURRENT_POINT_OPACITY);
```

```
                paint.setColor(resultPointColor);
                synchronized (currentPossible) {
                    for (ResultPoint point : currentPossible) {
                        canvas.drawCircle(frameLeft + (int) (point.getX() * scaleX), frameTop + (int)
(point.getY() * scaleY), POINT_SIZE, paint);
                    }
                }
            }
            if (currentLast != null) {
                paint.setAlpha(CURRENT_POINT_OPACITY / 2);
                paint.setColor(resultPointColor);
                synchronized (currentLast) {
                    for (ResultPoint point : currentLast) {
                        canvas.drawCircle(frameLeft + (int) (point.getX() * scaleX), frameTop + (int)
(point.getY() * scaleY), POINT_SIZE / 2, paint);
                    }
                }
            }

            // Request another update at the animation interval, but only
            // repaint the laser line,
            // not the entire viewfinder mask
            postInvalidateDelayed(ANIMATION_DELAY, frame.left - POINT_SIZE, frame.top - POINT_SIZE,
frame.right + POINT_SIZE, frame.bottom + POINT_SIZE);
        }
    }

    public void drawViewfinder() {
        Bitmap resultBitmap = this.resultBitmap;
        this.resultBitmap = null;
        if (resultBitmap != null) {
            resultBitmap.recycle();
        }
        invalidate();
    }

    /**
     *绘制位图与突出显示的视图，而不是实时扫描显示
     *
     * @param barcode
     *              An image of the decoded barcode.
     */
    public void drawResultBitmap(Bitmap barcode) {
        resultBitmap = barcode;
        invalidate();
    }

    public void addPossibleResultPoint(ResultPoint point) {
        List<ResultPoint> points = possibleResultPoints;
        synchronized (point) {
            points.add(point);
```

```
            int size = points.size();
            if (size > MAX_RESULT_POINTS) {
                // trim it
                points.subList(0, size - MAX_RESULT_POINTS / 2).clear();
            }
        }
    }
}
```

18.2 解码处理

知识点讲解：光盘:视频\知识点\第 18 章\解码处理.avi

当拍照采集到 QR 二维码信息后，需要通过解码机制来获取在这个二维码中包含的具体信息。在本节的内容中，将详细讲解解码功能的具体实现流程。

18.2.1 实现解码处理功能

编写文件 DecoderActivity.java，功能是调用并加载解码 UI 布局文件中的控件，具体实现代码如下所示。

```
public class DecoderActivity extends Activity implements IDecoderActivity, SurfaceHolder.Callback {
    private static final String TAG = DecoderActivity.class.getSimpleName();
    protected DecoderActivityHandler handler = null;
    protected ViewfinderView viewfinderView = null;
    protected CameraManager cameraManager = null;
    protected boolean hasSurface = false;
    protected Collection<BarcodeFormat> decodeFormats = null;
    protected String characterSet = null;
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.decoder);
        Log.v(TAG, "onCreate()");
        Window window = getWindow();
        window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        handler = null;
        hasSurface = false;
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.v(TAG, "onDestroy()");
    }

    @Override
    protected void onResume() {
        super.onResume();
```

```
        Log.v(TAG, "onResume()");

        // 这里的 CameraManager 必须初始化，而不是在 onCreate()中进行
        if (cameraManager == null) cameraManager = new CameraManager(getApplication());

        if (viewfinderView == null) {
            viewfinderView = (ViewfinderView) findViewById(R.id.viewfinder_view);
            viewfinderView.setCameraManager(cameraManager);
        }

        showScanner();

        SurfaceView surfaceView = (SurfaceView) findViewById(R.id.preview_view);
        SurfaceHolder surfaceHolder = surfaceView.getHolder();
        if (hasSurface) {
            // The activity was paused but not stopped, so the surface still
            // exists. Therefore
            // surfaceCreated() won't be called, so init the camera here
            initCamera(surfaceHolder);
        } else {
            // Install the callback and wait for surfaceCreated() to init the
            // camera
            surfaceHolder.addCallback(this);
            surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.v(TAG, "onPause()");

        if (handler != null) {
            handler.quitSynchronously();
            handler = null;
        }

        cameraManager.closeDriver();

        if (!hasSurface) {
            SurfaceView surfaceView = (SurfaceView) findViewById(R.id.preview_view);
            SurfaceHolder surfaceHolder = surfaceView.getHolder();
            surfaceHolder.removeCallback(this);
        }
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_FOCUS || keyCode == KeyEvent.KEYCODE_CAMERA) {
            //处理这些事件，使它们不启动相机应用程序
            return true;
```

```
        }
        return super.onKeyDown(keyCode, event);
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        if (holder == null)
            Log.e(TAG, "*** WARNING *** surfaceCreated() gave us a null surface!");
        if (!hasSurface) {
            hasSurface = true;
            initCamera(holder);
        }
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        hasSurface = false;
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
        // Ignore
    }

    @Override
    public ViewfinderView getViewfinder() {
        return viewfinderView;
    }

    @Override
    public Handler getHandler() {
        return handler;
    }

    @Override
    public CameraManager getCameraManager() {
        return cameraManager;
    }

    @Override
    public void handleDecode(Result rawResult, Bitmap barcode) {
        drawResultPoints(barcode, rawResult);
    }

    protected void drawResultPoints(Bitmap barcode, Result rawResult) {
        ResultPoint[ ] points = rawResult.getResultPoints();
        if (points != null && points.length > 0) {
            Canvas canvas = new Canvas(barcode);
            Paint paint = new Paint();
            paint.setColor(getResources().getColor(R.color.result_image_border));
            paint.setStrokeWidth(3.0f);
```

```
                paint.setStyle(Paint.Style.STROKE);
                Rect border = new Rect(2, 2, barcode.getWidth() - 2, barcode.getHeight() - 2);
                canvas.drawRect(border, paint);

                paint.setColor(getResources().getColor(R.color.result_points));
                if (points.length == 2) {
                     paint.setStrokeWidth(4.0f);
                     drawLine(canvas, paint, points[0], points[1]);
                } else if (points.length == 4 && (rawResult.getBarcodeFormat() == BarcodeFormat.UPC_A ||
rawResult.getBarcodeFormat() == BarcodeFormat.EAN_13)) {
                     //画出两条线，用于表示条形码和元数据
                     drawLine(canvas, paint, points[0], points[1]);
                     drawLine(canvas, paint, points[2], points[3]);
                } else {
                     paint.setStrokeWidth(10.0f);
                     for (ResultPoint point : points) {
                          canvas.drawPoint(point.getX(), point.getY(), paint);
                     }
                }
           }
      }

      protected static void drawLine(Canvas canvas, Paint paint, ResultPoint a, ResultPoint b) {
           canvas.drawLine(a.getX(), a.getY(), b.getX(), b.getY(), paint);
      }

      protected void showScanner() {
           viewfinderView.setVisibility(View.VISIBLE);
      }

      protected void initCamera(SurfaceHolder surfaceHolder) {
           try {
                cameraManager.openDriver(surfaceHolder);
                //创建处理程序开始预览，这也可以抛出一个 RuntimeException 异常
                if (handler == null) handler = new DecoderActivityHandler(this, decodeFormats, characterSet,
cameraManager);
           } catch (IOException ioe) {
                Log.w(TAG, ioe);
           } catch (RuntimeException e) {
                // Barcode Scanner has seen crashes in the wild of this variety:
                // java.?lang.?RuntimeException: Fail to connect to camera service
                Log.w(TAG, "Unexpected error initializing camera", e);
           }
      }
}
```

布局文件 decoder.xml 的具体实现代码如下所示。

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent">
```

```
    <SurfaceView android:id="@+id/preview_view"
                 android:layout_width="fill_parent"
                 android:layout_height="fill_parent">
    </SurfaceView>

    <com.jwetherell.quick_response_code.ViewfinderView
        android:id="@+id/viewfinder_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@color/transparent">
    </com.jwetherell.quick_response_code.ViewfinderView>

    <TextView android:id="@+id/status_view"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_gravity="bottom|center_horizontal"
              android:background="@color/transparent"
              android:text="@string/msg_default_status"
              android:textColor="@color/status_text"
              android:textSize="14sp"/>

</FrameLayout>
```

18.2.2 解码矩形框中的数据

编写文件 DecodeHandler.java，功能是获取取景器矩形框中的数据，并对矩形框区域中的数据进行解码处理。文件 DecodeHandler.java 的具体实现代码如下所示。

```
final class DecodeHandler extends Handler {
    private static final String TAG = DecodeHandler.class.getSimpleName();
    private final IDecoderActivity activity;
    private final MultiFormatReader multiFormatReader;
    private boolean running = true;
    DecodeHandler(IDecoderActivity activity, Map<DecodeHintType, Object> hints) {
        multiFormatReader = new MultiFormatReader();
        multiFormatReader.setHints(hints);
        this.activity = activity;
    }
    @Override
    public void handleMessage(Message message) {
        if (!running) {
            return;
        }
        switch (message.what) {
        case R.id.decode:
            decode((byte[ ]) message.obj, message.arg1, message.arg2);
            break;
        case R.id.quit:
            running = false;
            Looper.myLooper().quit();
            break;
```

```
        }
    }
    private void decode(byte[ ] data, int width, int height) {
        long start = System.currentTimeMillis();
        Result rawResult = null;
        PlanarYUVLuminanceSource   source   =   activity.getCameraManager().buildLuminanceSource(data,
width, height);
        if (source != null) {
            BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source));
            try {
                rawResult = multiFormatReader.decodeWithState(bitmap);
            } catch (ReaderException re) {
                // continue
            } finally {
                multiFormatReader.reset();
            }
        }

        Handler handler = activity.getHandler();
        if (rawResult != null) {
            // Don't log the barcode contents for security
            long end = System.currentTimeMillis();
            Log.d(TAG, "Found barcode in " + (end - start) + " ms");
            if (handler != null) {
                Message message = Message.obtain(handler, R.id.decode_succeeded, rawResult);
                Bundle bundle = new Bundle();
                bundle.putParcelable(DecodeThread.BARCODE_BITMAP, source.renderCroppedGreyscale-
Bitmap());
                message.setData(bundle);
                message.sendToTarget();
            }
        } else {
            if (handler != null) {
                Message message = Message.obtain(handler, R.id.decode_failed);
                message.sendToTarget();
            }
        }
    }
}
```

18.2.3　处理全部状态的采集信息

文件 DecoderActivityHandler.java 的功能是处理全部状态的采集信息，通过自动对焦技术迅速采集指定区域的 QR 信息。文件 DecoderActivityHandler.java 的具体实现代码如下所示。

```
public final class DecoderActivityHandler extends Handler {
    private static final String TAG = DecoderActivityHandler.class.getSimpleName();
    private final IDecoderActivity activity;
    private final DecodeThread decodeThread;
    private final CameraManager cameraManager;
    private State state;
```

```
    private enum State {
        PREVIEW, SUCCESS, DONE
    }
    DecoderActivityHandler(IDecoderActivity activity, Collection<BarcodeFormat> decodeFormats, String
characterSet,
            CameraManager cameraManager) {
        this.activity = activity;
        decodeThread = new DecodeThread(activity, decodeFormats, characterSet, new ViewfinderResult
PointCallback(
                activity.getViewfinder()));
        decodeThread.start();
        state = State.SUCCESS;

        // Start ourselves capturing previews and decoding
        this.cameraManager = cameraManager;
        cameraManager.startPreview();
        restartPreviewAndDecode();
    }

    @Override
    public void handleMessage(Message message) {
        switch (message.what) {
            case R.id.auto_focus:
                // Log.d(TAG, "Got auto-focus message");
                //实现自动对焦，原理是当一个自动对焦完成时马上开始另一个
                if (state == State.PREVIEW) cameraManager.requestAutoFocus(this, R.id.auto_focus);
                break;
            case R.id.restart_preview:
                Log.d(TAG, "Got restart preview message");
                restartPreviewAndDecode();
                break;
            case R.id.decode_succeeded:
                Log.d(TAG, "Got decode succeeded message");
                state = State.SUCCESS;
                Bundle bundle = message.getData();
                Bitmap barcode = bundle == null ? null : (Bitmap) bundle.getParcelable(DecodeThread.
BARCODE_BITMAP);
                activity.handleDecode((Result) message.obj, barcode);
                break;
            case R.id.decode_failed:
                //快速解码，当一个解码失败后启动另一个
                state = State.PREVIEW;
                cameraManager.requestPreviewFrame(decodeThread.getHandler(), R.id.decode);
                break;
            case R.id.return_scan_result:
                Log.d(TAG, "Got return scan result message");
                if (activity instanceof Activity) {
                    ((Activity) activity).setResult(Activity.RESULT_OK, (Intent) message.obj);
                    ((Activity) activity).finish();
                } else {
                    Log.e(TAG, "Scan result message, activity is not Activity. Doing nothing.");
```

```
                }
                break;
            }
        }

        public void quitSynchronously() {
            state = State.DONE;
            cameraManager.stopPreview();
            Message quit = Message.obtain(decodeThread.getHandler(), R.id.quit);
            quit.sendToTarget();
            try {
                //最多等待半秒钟，在 onPause()时将很快超时
                decodeThread.join(500L);
            } catch (InterruptedException e) {
                // continue
            }

            //绝对确保不会发送任何排队的消息
            removeMessages(R.id.decode_succeeded);
            removeMessages(R.id.decode_failed);
        }

        void restartPreviewAndDecode() {
            if (state == State.SUCCESS) {
                state = State.PREVIEW;
                cameraManager.requestPreviewFrame(decodeThread.getHandler(), R.id.decode);
                cameraManager.requestAutoFocus(this, R.id.auto_focus);
                activity.getViewfinder().drawViewfinder();
            }
        }
}
```

18.2.4　多线程处理

为了提高解码效率，编写文件 DecodeThread.java，功能是快速执行图像解码工作，具体实现代码如下所示。

```
final class DecodeThread extends Thread {

        private static final String TAG = DecodeThread.class.getSimpleName();
        public static final String BARCODE_BITMAP = "barcode_bitmap";

        private final IDecoderActivity activity;
        private final Map<DecodeHintType, Object> hints;
        private final CountDownLatch handlerInitLatch;

        private Handler handler;

        DecodeThread(IDecoderActivity activity, Collection<BarcodeFormat> decodeFormats, String characterSet,
                ResultPointCallback resultPointCallback) {
```

```
        this.activity = activity;
        handlerInitLatch = new CountDownLatch(1);
        hints = new EnumMap<DecodeHintType, Object>(DecodeHintType.class);

        // The prefs can't change while the thread is running, so pick them up
        // once here
        if (decodeFormats == null || decodeFormats.isEmpty()) {
            if (activity instanceof Activity) {
                decodeFormats = EnumSet.noneOf(BarcodeFormat.class);
                if (Preferences.KEY_DECODE_1D) {
                    decodeFormats.addAll(DecodeFormatManager.ONE_D_FORMATS);
                }
                if (Preferences.KEY_DECODE_QR) {
                    decodeFormats.addAll(DecodeFormatManager.QR_CODE_FORMATS);
                }
                if (Preferences.KEY_DECODE_DATA_MATRIX) {
                    decodeFormats.addAll(DecodeFormatManager.DATA_MATRIX_FORMATS);
                }
            } else {
                Log.e(TAG, "activity is not an Activity, not handling preferences.");
            }
        }
        hints.put(DecodeHintType.POSSIBLE_FORMATS, decodeFormats);
        if (characterSet != null) {
            hints.put(DecodeHintType.CHARACTER_SET, characterSet);
        }
        hints.put(DecodeHintType.NEED_RESULT_POINT_CALLBACK, resultPointCallback);
    }
    Handler getHandler() {
        try {
            handlerInitLatch.await();
        } catch (InterruptedException ie) {
            // continue?
        }
        return handler;
    }
    @Override
    public void run() {
        Looper.prepare();
        handler = new DecodeHandler(activity, hints);
        handlerInitLatch.countDown();
        Looper.loop();
    }
}
```

18.2.5　读取 QR 码

编写文件 QRCodeReader.java 读取 QR 二维码的信息，通过此文件可以检测并读取图像中的 QR 二维码信息。文件 QRCodeReader.java 的具体实现代码如下所示。

```
public class QRCodeReader implements Reader {
    private static final ResultPoint[ ] NO_POINTS = new ResultPoint[0];
    private final Decoder decoder = new Decoder();
    protected Decoder getDecoder() {
        return decoder;
    }
    /**
     *定位和图像解码的 QR 码
     */
    @Override
    public Result decode(BinaryBitmap image) throws NotFoundException, ChecksumException,
FormatException {
        return decode(image, null);
    }
    @Override
    public Result decode(BinaryBitmap image, Map<DecodeHintType, ?> hints) throws NotFoundException,
ChecksumException, FormatException {
        DecoderResult decoderResult;
        ResultPoint[ ] points;
        if (hints != null && hints.containsKey(DecodeHintType.PURE_BARCODE)) {
            BitMatrix bits = extractPureBits(image.getBlackMatrix());
            decoderResult = decoder.decode(bits, hints);
            points = NO_POINTS;
        } else {
            DetectorResult detectorResult = new Detector(image.getBlackMatrix()).detect(hints);
            decoderResult = decoder.decode(detectorResult.getBits(), hints);
            points = detectorResult.getPoints();
        }
        Result result = new Result(decoderResult.getText(), decoderResult.getRawBytes(), points,
BarcodeFormat.QR_CODE);
        List<byte[ ]> byteSegments = decoderResult.getByteSegments();
        if (byteSegments != null) {
            result.putMetadata(ResultMetadataType.BYTE_SEGMENTS, byteSegments);
        }
        String ecLevel = decoderResult.getECLevel();
        if (ecLevel != null) {
            result.putMetadata(ResultMetadataType.ERROR_CORRECTION_LEVEL, ecLevel);
        }
        return result;
    }
    @Override
    public void reset() {
        // do nothing
    }
```

```
private static BitMatrix extractPureBits(BitMatrix image) throws NotFoundException {
    int[ ] leftTopBlack = image.getTopLeftOnBit();
    int[ ] rightBottomBlack = image.getBottomRightOnBit();
    if (leftTopBlack == null || rightBottomBlack == null) {
        throw NotFoundException.getNotFoundInstance();
    }

    int moduleSize = moduleSize(leftTopBlack, image);

    int top = leftTopBlack[1];
    int bottom = rightBottomBlack[1];
    int left = leftTopBlack[0];
    int right = rightBottomBlack[0];

    if (bottom - top != right - left) {
        //特殊情况下，在右下角的模块是不是黑色的
        right = left + (bottom - top);
    }

    int matrixWidth = (right - left + 1) / moduleSize;
    int matrixHeight = (bottom - top + 1) / moduleSize;
    if (matrixWidth <= 0 || matrixHeight <= 0) {
        throw NotFoundException.getNotFoundInstance();
    }
    if (matrixHeight != matrixWidth) {
        // Only possibly decode square regions
        throw NotFoundException.getNotFoundInstance();
    }
    //短暂关闭，这将有助于恢复
    int nudge = moduleSize >> 1;
    top += nudge;
    left += nudge;
    //只要读出位
    BitMatrix bits = new BitMatrix(matrixWidth, matrixHeight);
    for (int y = 0; y < matrixHeight; y++) {
        int iOffset = top + y * moduleSize;
        for (int x = 0; x < matrixWidth; x++) {
            if (image.get(left + x * moduleSize, iOffset)) {
                bits.set(x, y);
            }
        }
    }
    return bits;
}
private static int moduleSize(int[ ] leftTopBlack, BitMatrix image) throws NotFoundException {
    int height = image.getHeight();
    int width = image.getWidth();
    int x = leftTopBlack[0];
    int y = leftTopBlack[1];
    while (x < width && y < height && image.get(x, y)) {
        x++;
```

```
            y++;
        }
        if (x == width || y == height) {
            throw NotFoundException.getNotFoundInstance();
        }
        int moduleSize = x - leftTopBlack[0];
        if (moduleSize == 0) {
            throw NotFoundException.getNotFoundInstance();
        }
        return moduleSize;
    }
}
```

18.3　编 码 处 理

知识点讲解：光盘:视频\知识点\第 18 章\编码处理.avi

编码处理的过程和解码处理的过程相反，是指将采集到的信息编码成 QR 二维码的形式。在本节的内容中，将详细讲解编码处理的具体实现流程。

18.3.1　Encoder 处理

编写文件 EncoderActivity.java，功能是使用 Encoder 将采集到的信息转换为 QR 码，具体实现代码如下所示。

```
public final class EncoderActivity extends Activity {

    private static final String TAG = EncoderActivity.class.getSimpleName();

    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.encoder);

        //假设全屏视图时的操作
        WindowManager manager = (WindowManager) getSystemService(WINDOW_SERVICE);
        Display display = manager.getDefaultDisplay();
        int width = display.getWidth();
        int height = display.getHeight();
        int smallerDimension = width < height ? width : height;
        smallerDimension = smallerDimension * 7 / 8;

        try {
            QRCodeEncoder qrCodeEncoder = null;
            // qrCodeEncoder = new QRCodeEncoder("AT", null, Contents.Type.TEXT,
            // BarcodeFormat.CODABAR.toString(), smallerDimension);
            // qrCodeEncoder = new QRCodeEncoder("HI", null, Contents.Type.TEXT,
            // BarcodeFormat.CODE_39.toString(), smallerDimension);
            // qrCodeEncoder = new QRCodeEncoder("Hello", null,
```

```
                // Contents.Type.TEXT, BarcodeFormat.CODE_128.toString(),
                // smallerDimension);
                // qrCodeEncoder = new QRCodeEncoder("1234567891011", null,
                // Contents.Type.TEXT, BarcodeFormat.EAN_13.toString(),
                // smallerDimension);
                // qrCodeEncoder = new QRCodeEncoder("12345678", null,
                // Contents.Type.TEXT, BarcodeFormat.EAN_8.toString(),
                // smallerDimension);
                // qrCodeEncoder = new QRCodeEncoder("1234", null,
                // Contents.Type.TEXT, BarcodeFormat.ITF.toString(),
                // smallerDimension);
                // qrCodeEncoder = new QRCodeEncoder("2345", null,
                // Contents.Type.TEXT, BarcodeFormat.PDF_417.toString(),
                // smallerDimension);
                qrCodeEncoder = new QRCodeEncoder("Hello", null, Contents.Type.TEXT, BarcodeFormat.QR_
CODE.toString(), smallerDimension);
                // qrCodeEncoder = new QRCodeEncoder("12345678910", null,
                // Contents.Type.TEXT, BarcodeFormat.UPC_A.toString(),
                // smallerDimension);

                Bitmap bitmap = qrCodeEncoder.encodeAsBitmap();
                ImageView view = (ImageView) findViewById(R.id.image_view);
                view.setImageBitmap(bitmap);

                TextView contents = (TextView) findViewById(R.id.contents_text_view);
                contents.setText(qrCodeEncoder.getDisplayContents());
                setTitle(getString(R.string.app_name) + " - " + qrCodeEncoder.getTitle());
            } catch (WriterException e) {
                Log.e(TAG, "Could not encode barcode", e);
            } catch (IllegalArgumentException e) {
                Log.e(TAG, "Could not encode barcode", e);
            }
        }
}
```

在上述代码中加载显示了布局文件 encoder.xml 中的 UI 控件，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                  android:id="@+id/encode_view"
                  android:layout_width="fill_parent"
                  android:layout_height="fill_parent"
                  android:fillViewport="true"
                  android:background="@color/encode_view"
                  android:orientation="vertical"
                  android:gravity="center">
  <ImageView android:id="@+id/image_view"
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:layout_gravity="center_horizontal"
                  android:scaleType="center"
                  android:contentDescription="@string/barcode_image"/>
  <ScrollView  android:layout_width="fill_parent"
```

```
                    android:layout_height="wrap_content"
                    android:layout_gravity="center_horizontal"
                    android:gravity="center">
        <TextView android:id="@+id/contents_text_view"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center_horizontal"
                    android:gravity="center"
                    android:textColor="@color/contents_text"
                    android:textSize="18sp"
                    android:paddingBottom="8dip"
                    android:paddingLeft="8dip"
                    android:paddingRight="8dip"/>
    </ScrollView>
</LinearLayout>
```

18.3.2 生成 QR 二维码

编写文件 Encoder.java，功能是以指定的编码模式生成 QR 二维码。可以在 chooseMode 内部选择编码模式，编码成功后被存储在 QRCode 结果集中。在此建议使用 QRCode.EC_LEVEL_L（最低级）模式，因为我们的主要用途是显示 QR 码上的桌面屏幕。如果需要非常强的纠错功能，可以选择其他的模式。文件 Encoder.java 的具体实现代码如下所示。

```
public final class Encoder {
    // The original table is defined in the table 5 of JISX0510:2004 (p.19).
    private static final int[ ] ALPHANUMERIC_TABLE = { -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
// 0x00-0x0f
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, // 0x10-0x1f
            36, -1, -1, -1, 37, 38, -1, -1, -1, -1, 39, 40, -1, 41, 42, 43, // 0x20-0x2f
            0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 44, -1, -1, -1, -1, -1, // 0x30-0x3f
            -1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, // 0x40-0x4f
            25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, -1, -1, -1, -1, -1, // 0x50-0x5f
    };

    static final String DEFAULT_BYTE_MODE_ENCODING = "ISO-8859-1";

    private Encoder() {
    }

    // The mask penalty calculation is complicated. See Table 21 of
    // JISX0510:2004 (p.45) for
    // details.
    // Basically it applies four rules and summate all penalties.
    private static int calculateMaskPenalty(ByteMatrix matrix) {
        int penalty = 0;
        penalty += MaskUtil.applyMaskPenaltyRule1(matrix);
        penalty += MaskUtil.applyMaskPenaltyRule2(matrix);
        penalty += MaskUtil.applyMaskPenaltyRule3(matrix);
        penalty += MaskUtil.applyMaskPenaltyRule4(matrix);
        return penalty;
```

```
    }
    public static void encode(String content, ErrorCorrectionLevel ecLevel, QRCode qrCode) throws
WriterException {
        encode(content, ecLevel, null, qrCode);
    }

    public static void encode(String content, ErrorCorrectionLevel ecLevel, Map<EncodeHintType, ?> hints,
QRCode qrCode) throws WriterException {

        String encoding = hints == null ? null : (String) hints.get(EncodeHintType.CHARACTER_SET);
        if (encoding == null) {
            encoding = DEFAULT_BYTE_MODE_ENCODING;
        }

        // Step 1: Choose the mode (encoding)
        Mode mode = chooseMode(content, encoding);

        // Step 2: Append "bytes" into "dataBits" in appropriate encoding
        BitArray dataBits = new BitArray();
        appendBytes(content, mode, dataBits, encoding);

        // Step 3: Initialize QR code that can contain "dataBits"
        int numInputBits = dataBits.getSize();
        initQRCode(numInputBits, ecLevel, mode, qrCode);

        // Step 4: Build another bit vector that contains header and data
        BitArray headerAndDataBits = new BitArray();

        // Step 4.5: Append ECI message if applicable
        if (mode == Mode.BYTE && !DEFAULT_BYTE_MODE_ENCODING.equals(encoding)) {
            CharacterSetECI eci = CharacterSetECI.getCharacterSetECIByName(encoding);
            if (eci != null) {
                appendECI(eci, headerAndDataBits);
            }
        }

        appendModeInfo(mode, headerAndDataBits);

        int numLetters = mode == Mode.BYTE ? dataBits.getSizeInBytes() : content.length();
        appendLengthInfo(numLetters, qrCode.getVersion(), mode, headerAndDataBits);
        headerAndDataBits.appendBitArray(dataBits);

        // Step 5: Terminate the bits properly
        terminateBits(qrCode.getNumDataBytes(), headerAndDataBits);

        // Step 6: Interleave data bits with error correction code
        BitArray finalBits = new BitArray();
        interleaveWithECBytes(headerAndDataBits, qrCode.getNumTotalBytes(), qrCode.getNumDataBytes(),
qrCode.getNumRSBlocks(), finalBits);

        // Step 7: Choose the mask pattern and set to "qrCode"
```

```
        ByteMatrix matrix = new ByteMatrix(qrCode.getMatrixWidth(), qrCode.getMatrixWidth());
        qrCode.setMaskPattern(chooseMaskPattern(finalBits, ecLevel, qrCode.getVersion(), matrix));

        // Step 8. Build the matrix and set it to "qrCode"
        MatrixUtil.buildMatrix(finalBits, ecLevel, qrCode.getVersion(), qrCode.getMaskPattern(), matrix);
        qrCode.setMatrix(matrix);

        // Step 9. Make sure we have a valid QR Code
        if (!qrCode.isValid()) {
            throw new WriterException("Invalid QR code: " + qrCode.toString());
        }
    }

    static int getAlphanumericCode(int code) {
        if (code < ALPHANUMERIC_TABLE.length) {
            return ALPHANUMERIC_TABLE[code];
        }
        return -1;
    }

    public static Mode chooseMode(String content) {
        return chooseMode(content, null);
    }
    private static Mode chooseMode(String content, String encoding) {
        if ("Shift_JIS".equals(encoding)) {
            // Choose Kanji mode if all input are double-byte characters
            return isOnlyDoubleByteKanji(content) ? Mode.KANJI : Mode.BYTE;
        }
        boolean hasNumeric = false;
        boolean hasAlphanumeric = false;
        for (int i = 0; i < content.length(); ++i) {
            char c = content.charAt(i);
            if (c >= '0' && c <= '9') {
                hasNumeric = true;
            } else if (getAlphanumericCode(c) != -1) {
                hasAlphanumeric = true;
            } else {
                return Mode.BYTE;
            }
        }
        if (hasAlphanumeric) {
            return Mode.ALPHANUMERIC;
        }
        if (hasNumeric) {
            return Mode.NUMERIC;
        }
        return Mode.BYTE;
    }

    private static boolean isOnlyDoubleByteKanji(String content) {
        byte[ ] bytes;
```

```
            try {
                  bytes = content.getBytes("Shift_JIS");
            } catch (UnsupportedEncodingException uee) {
                  return false;
            }
            int length = bytes.length;
            if (length % 2 != 0) {
                  return false;
            }
            for (int i = 0; i < length; i += 2) {
                  int byte1 = bytes[i] & 0xFF;
                  if ((byte1 < 0x81 || byte1 > 0x9F) && (byte1 < 0xE0 || byte1 > 0xEB)) {
                        return false;
                  }
            }
            return true;
      }

      private static int chooseMaskPattern(BitArray bits, ErrorCorrectionLevel ecLevel, int version, ByteMatrix
matrix) throws WriterException {

            int minPenalty = Integer.MAX_VALUE; // Lower penalty is better.
            int bestMaskPattern = -1;
            // We try all mask patterns to choose the best one
            for (int maskPattern = 0; maskPattern < QRCode.NUM_MASK_PATTERNS; maskPattern++) {
                  MatrixUtil.buildMatrix(bits, ecLevel, version, maskPattern, matrix);
                  int penalty = calculateMaskPenalty(matrix);
                  if (penalty < minPenalty) {
                        minPenalty = penalty;
                        bestMaskPattern = maskPattern;
                  }
            }
            return bestMaskPattern;
      }
      private static void initQRCode(int numInputBits, ErrorCorrectionLevel ecLevel, Mode mode, QRCode
qrCode) throws WriterException {
            qrCode.setECLevel(ecLevel);
            qrCode.setMode(mode);

            // In the following comments, we use numbers of Version 7-H
            for (int versionNum = 1; versionNum <= 40; versionNum++) {
                  Version version = Version.getVersionForNumber(versionNum);
                  // numBytes = 196
                  int numBytes = version.getTotalCodewords();
                  // getNumECBytes = 130
                  Version.ECBlocks ecBlocks = version.getECBlocksForLevel(ecLevel);
                  int numEcBytes = ecBlocks.getTotalECCodewords();
                  // getNumRSBlocks = 5
                  int numRSBlocks = ecBlocks.getNumBlocks();
                  // getNumDataBytes = 196 - 130 = 66
                  int numDataBytes = numBytes - numEcBytes;
```

```
            // We want to choose the smallest version which can contain data of
            // "numInputBytes" +
            // some
            // extra bits for the header (mode info and length info). The header
            // can be three bytes
            // (precisely 4 + 16 bits) at most
            if (numDataBytes >= getTotalInputBytes(numInputBits, version, mode)) {
                // Yay, we found the proper rs block info!
                qrCode.setVersion(versionNum);
                qrCode.setNumTotalBytes(numBytes);
                qrCode.setNumDataBytes(numDataBytes);
                qrCode.setNumRSBlocks(numRSBlocks);
                // getNumECBytes = 196 - 66 = 130
                qrCode.setNumECBytes(numEcBytes);
                // matrix width = 21 + 6 * 4 = 45
                qrCode.setMatrixWidth(version.getDimensionForVersion());
                return;
            }
        }
        throw new WriterException("Cannot find proper rs block info (input data too big?)");
    }

    private static int getTotalInputBytes(int numInputBits, Version version, Mode mode) {
        int modeInfoBits = 4;
        int charCountBits = mode.getCharacterCountBits(version);
        int headerBits = modeInfoBits + charCountBits;
        int totalBits = numInputBits + headerBits;

        return (totalBits + 7) / 8;
    }

    /**
     * 终止位在 8.4.8 和 8.4.9 之间
     */
    static void terminateBits(int numDataBytes, BitArray bits) throws WriterException {
        int capacity = numDataBytes << 3;
        if (bits.getSize() > capacity) {
            throw new WriterException("data bits cannot fit in the QR Code" + bits.getSize() + " > " + capacity);
        }
        for (int i = 0; i < 4 && bits.getSize() < capacity; ++i) {
            bits.appendBit(false);
        }
        // Append termination bits. See 8.4.8 of JISX0510:2004 (p.24) for
        // details.
        // If the last byte isn't 8-bit aligned, we'll add padding bits.
        int numBitsInLastByte = bits.getSize() & 0x07;
        if (numBitsInLastByte > 0) {
            for (int i = numBitsInLastByte; i < 8; i++) {
                bits.appendBit(false);
            }
        }
```

```
        // If we have more space, we'll fill the space with padding patterns
        // defined in 8.4.9
        // (p.24).
        int numPaddingBytes = numDataBytes - bits.getSizeInBytes();
        for (int i = 0; i < numPaddingBytes; ++i) {
            bits.appendBits((i & 0x01) == 0 ? 0xEC : 0x11, 8);
        }
        if (bits.getSize() != capacity) {
            throw new WriterException("Bits size does not equal capacity");
        }
    }

    /**
     *得到的数据字节和纠错的数字节数块
     */
    static void getNumDataBytesAndNumECBytesForBlockID(int numTotalBytes, int numDataBytes, int
numRSBlocks, int blockID, int[ ] numDataBytesInBlock,
            int[ ] numECBytesInBlock) throws WriterException {
        if (blockID >= numRSBlocks) {
            throw new WriterException("Block ID too large");
        }
        // numRsBlocksInGroup2 = 196 % 5 = 1
        int numRsBlocksInGroup2 = numTotalBytes % numRSBlocks;
        // numRsBlocksInGroup1 = 5 - 1 = 4
        int numRsBlocksInGroup1 = numRSBlocks - numRsBlocksInGroup2;
        // numTotalBytesInGroup1 = 196 / 5 = 39
        int numTotalBytesInGroup1 = numTotalBytes / numRSBlocks;
        // numTotalBytesInGroup2 = 39 + 1 = 40
        int numTotalBytesInGroup2 = numTotalBytesInGroup1 + 1;
        // numDataBytesInGroup1 = 66 / 5 = 13
        int numDataBytesInGroup1 = numDataBytes / numRSBlocks;
        // numDataBytesInGroup2 = 13 + 1 = 14
        int numDataBytesInGroup2 = numDataBytesInGroup1 + 1;
        // numEcBytesInGroup1 = 39 - 13 = 26
        int numEcBytesInGroup1 = numTotalBytesInGroup1 - numDataBytesInGroup1;
        // numEcBytesInGroup2 = 40 - 14 = 26
        int numEcBytesInGroup2 = numTotalBytesInGroup2 - numDataBytesInGroup2;
        // Sanity checks.
        // 26 = 26
        if (numEcBytesInGroup1 != numEcBytesInGroup2) {
            throw new WriterException("EC bytes mismatch");
        }
        // 5 = 4 + 1
        if (numRSBlocks != numRsBlocksInGroup1 + numRsBlocksInGroup2) {
            throw new WriterException("RS blocks mismatch");
        }
        // 196 = (13 + 26) * 4 + (14 + 26) * 1
        if (numTotalBytes != ((numDataBytesInGroup1 + numEcBytesInGroup1) * numRsBlocksInGroup1)
                + ((numDataBytesInGroup2 + numEcBytesInGroup2) * numRsBlocksInGroup2)) {
            throw new WriterException("Total bytes mismatch");
        }
```

```
        if (blockID < numRsBlocksInGroup1) {
            numDataBytesInBlock[0] = numDataBytesInGroup1;
            numECBytesInBlock[0] = numEcBytesInGroup1;
        } else {
            numDataBytesInBlock[0] = numDataBytesInGroup2;
            numECBytesInBlock[0] = numEcBytesInGroup2;
        }
    }
    static void interleaveWithECBytes(BitArray bits, int numTotalBytes, int numDataBytes, int numRSBlocks,
BitArray result) throws WriterException {

        // "bits" must have "getNumDataBytes" bytes of data
        if (bits.getSizeInBytes() != numDataBytes) {
            throw new WriterException("Number of bits and data bytes does not match");
        }

        //分割数据字节成块，并为它们生成纠错字节
        //将存储划分字节的数据块和纠错字节块存储到“块”
        int dataBytesOffset = 0;
        int maxNumDataBytes = 0;
        int maxNumEcBytes = 0;

        //根据 reedsolmon 块的数量值来初始化向量
        Collection<BlockPair> blocks = new ArrayList<BlockPair>(numRSBlocks);

        for (int i = 0; i < numRSBlocks; ++i) {
            int[ ] numDataBytesInBlock = new int[1];
            int[ ] numEcBytesInBlock = new int[1];
            getNumDataBytesAndNumECBytesForBlockID(numTotalBytes, numDataBytes, numRSBlocks, i,
numDataBytesInBlock, numEcBytesInBlock);

            int size = numDataBytesInBlock[0];
            byte[ ] dataBytes = new byte[size];
            bits.toBytes(8 * dataBytesOffset, dataBytes, 0, size);
            byte[ ] ecBytes = generateECBytes(dataBytes, numEcBytesInBlock[0]);
            blocks.add(new BlockPair(dataBytes, ecBytes));

            maxNumDataBytes = Math.max(maxNumDataBytes, size);
            maxNumEcBytes = Math.max(maxNumEcBytes, ecBytes.length);
            dataBytesOffset += numDataBytesInBlock[0];
        }
        if (numDataBytes != dataBytesOffset) {
            throw new WriterException("Data bytes does not match offset");
        }

        //数据块处理
        for (int i = 0; i < maxNumDataBytes; ++i) {
            for (BlockPair block : blocks) {
                byte[ ] dataBytes = block.getDataBytes();
                if (i < dataBytes.length) {
```

```
                    result.appendBits(dataBytes[i], 8);
                }
            }
        }
        //纠错块处理
        for (int i = 0; i < maxNumEcBytes; ++i) {
            for (BlockPair block : blocks) {
                byte[ ] ecBytes = block.getErrorCorrectionBytes();
                if (i < ecBytes.length) {
                    result.appendBits(ecBytes[i], 8);
                }
            }
        }
        if (numTotalBytes != result.getSizeInBytes()) { // Should be same.
            throw new WriterException("Interleaving error: " + numTotalBytes + " and " + result.get
SizeInBytes() + " differ.");
        }
    }
    static byte[ ] generateECBytes(byte[ ] dataBytes, int numEcBytesInBlock) {
        int numDataBytes = dataBytes.length;
        int[ ] toEncode = new int[numDataBytes + numEcBytesInBlock];
        for (int i = 0; i < numDataBytes; i++) {
            toEncode[i] = dataBytes[i] & 0xFF;
        }
        new ReedSolomonEncoder(GenericGF.QR_CODE_FIELD_256).encode(toEncode, numEcBytesInBlock);
        byte[ ] ecBytes = new byte[numEcBytesInBlock];
        for (int i = 0; i < numEcBytesInBlock; i++) {
            ecBytes[i] = (byte) toEncode[numDataBytes + i];
        }
        return ecBytes;
    }
    /**
     *追加模式信息。如果成功，将结果存储在“位”
     */
    static void appendModeInfo(Mode mode, BitArray bits) {
        bits.appendBits(mode.getBits(), 4);
    }
    /**
     *追加长度的信息。成功时，存储在“位”的结果
     */
    static void appendLengthInfo(int numLetters, int version, Mode mode, BitArray bits) throws WriterException {
        int numBits = mode.getCharacterCountBits(Version.getVersionForNumber(version));
        if (numLetters > ((1 << numBits) - 1)) {
            throw new WriterException(numLetters + "is bigger than" + ((1 << numBits) - 1));
        }
        bits.appendBits(numLetters, numBits);
    }

    /**
     *在“模式”模式（编码）中追加“字节”为“比特”。如果成功，则将结果存储在“位”
     */
```

```
    static void appendBytes(String content, Mode mode, BitArray bits, String encoding) throws WriterException
{
        switch (mode) {
        case NUMERIC:
            appendNumericBytes(content, bits);
            break;
        case ALPHANUMERIC:
            appendAlphanumericBytes(content, bits);
            break;
        case BYTE:
            append8BitBytes(content, bits, encoding);
            break;
        case KANJI:
            appendKanjiBytes(content, bits);
            break;
        default:
            throw new WriterException("Invalid mode: " + mode);
        }
    }

    static void appendNumericBytes(CharSequence content, BitArray bits) {
        int length = content.length();
        int i = 0;
        while (i < length) {
            int num1 = content.charAt(i) - '0';
            if (i + 2 < length) {
                //10 位编码的三个数字字母
                int num2 = content.charAt(i + 1) - '0';
                int num3 = content.charAt(i + 2) - '0';
                bits.appendBits(num1 * 100 + num2 * 10 + num3, 10);
                i += 3;
            } else if (i + 1 < length) {
                // 7 位编码的两个数字字母
                int num2 = content.charAt(i + 1) - '0';
                bits.appendBits(num1 * 10 + num2, 7);
                i += 2;
            } else {
                // Encode one numeric letter in four bits
                bits.appendBits(num1, 4);
                i++;
            }
        }
    }

    static void appendAlphanumericBytes(CharSequence content, BitArray bits) throws WriterException {
        int length = content.length();
        int i = 0;
        while (i < length) {
            int code1 = getAlphanumericCode(content.charAt(i));
            if (code1 == -1) {
                throw new WriterException();
```

```
            }
            if (i + 1 < length) {
                int code2 = getAlphanumericCode(content.charAt(i + 1));
                if (code2 == -1) {
                    throw new WriterException();
                }
                // Encode two alphanumeric letters in 11 bits
                bits.appendBits(code1 * 45 + code2, 11);
                i += 2;
            } else {
                // Encode one alphanumeric letter in six bits
                bits.appendBits(code1, 6);
                i++;
            }
        }
    }

    static void append8BitBytes(String content, BitArray bits, String encoding) throws WriterException {
        byte[ ] bytes;
        try {
            bytes = content.getBytes(encoding);
        } catch (UnsupportedEncodingException uee) {
            throw new WriterException(uee.toString());
        }
        for (byte b : bytes) {
            bits.appendBits(b, 8);
        }
    }

    static void appendKanjiBytes(String content, BitArray bits) throws WriterException {
        byte[ ] bytes;
        try {
            bytes = content.getBytes("Shift_JIS");
        } catch (UnsupportedEncodingException uee) {
            throw new WriterException(uee.toString());
        }
        int length = bytes.length;
        for (int i = 0; i < length; i += 2) {
            int byte1 = bytes[i] & 0xFF;
            int byte2 = bytes[i + 1] & 0xFF;
            int code = (byte1 << 8) | byte2;
            int subtracted = -1;
            if (code >= 0x8140 && code <= 0x9ffc) {
                subtracted = code - 0x8140;
            } else if (code >= 0xe040 && code <= 0xebbf) {
                subtracted = code - 0xc140;
            }
            if (subtracted == -1) {
                throw new WriterException("Invalid byte sequence");
            }
            int encoded = ((subtracted >> 8) * 0xc0) + (subtracted & 0xff);
```

```
            bits.appendBits(encoded, 13);
        }
    }
    private static void appendECI(CharacterSetECI eci, BitArray bits) {
        bits.appendBits(Mode.ECI.getBits(), 4);
        //正确的值高达 127
        bits.appendBits(eci.getValue(), 8);
    }
}
```

18.4　信 息 分 享

知识点讲解：光盘:视频\知识点\第 18 章\信息分享.avi

当读取 QR 码中的信息后，可以向里面添加联系信息，并将生成的 QR 二维码与邮箱、网站、推特或某个应用程序相连接。在本节的内容中，将详细讲解信息分享功能的具体实现流程。

18.4.1　通讯录处理

编写文件 AddressBookResultHandler.java 处理通讯录的信息，即处理 QR 码的地址项信息，具体实现代码如下所示。

```
public final class AddressBookResultHandler extends ResultHandler {

    private static final DateFormat[ ] DATE_FORMATS = { new SimpleDateFormat("yyyyMMdd", Locale.ENGLISH),
            new SimpleDateFormat("yyyyMMdd'T'HHmmss", Locale.ENGLISH), new SimpleDateFormat
("yyyy-MM-dd", Locale.ENGLISH),
            new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss", Locale.ENGLISH), };

    private final boolean[ ] fields;

    public AddressBookResultHandler(Activity activity, ParsedResult result) {
        super(activity, result);
        AddressBookParsedResult addressResult = (AddressBookParsedResult) result;
        String[ ] addresses = addressResult.getAddresses();
        boolean hasAddress = addresses != null && addresses.length > 0 && addresses[0].length() > 0;
        String[ ] phoneNumbers = addressResult.getPhoneNumbers();
        boolean hasPhoneNumber = phoneNumbers != null && phoneNumbers.length > 0;
        String[ ] emails = addressResult.getEmails();
        boolean hasEmailAddress = emails != null && emails.length > 0;

        fields = new boolean[4];
        fields[0] = true; // Add contact is always available
        fields[1] = hasAddress;
        fields[2] = hasPhoneNumber;
        fields[3] = hasEmailAddress;
    }

    //重写操作，可以判断是电话号码、生日和非常规的名字等格式
```

```
@Override
public CharSequence getDisplayContents() {
    AddressBookParsedResult result = (AddressBookParsedResult) getResult();
    StringBuilder contents = new StringBuilder(100);
    ParsedResult.maybeAppend(result.getNames(), contents);
    int namesLength = contents.length();

    String pronunciation = result.getPronunciation();
    if (pronunciation != null && pronunciation.length() > 0) {
        contents.append("\n(");
        contents.append(pronunciation);
        contents.append(')');
    }

    ParsedResult.maybeAppend(result.getTitle(), contents);
    ParsedResult.maybeAppend(result.getOrg(), contents);
    ParsedResult.maybeAppend(result.getAddresses(), contents);
    String[ ] numbers = result.getPhoneNumbers();
    if (numbers != null) {
        for (String number : numbers) {
            ParsedResult.maybeAppend(PhoneNumberUtils.formatNumber(number), contents);
        }
    }
    ParsedResult.maybeAppend(result.getEmails(), contents);
    ParsedResult.maybeAppend(result.getURL(), contents);

    String birthday = result.getBirthday();
    if (birthday != null && birthday.length() > 0) {
        Date date = parseDate(birthday);
        if (date != null) {
            ParsedResult.maybeAppend(DateFormat.getDateInstance().format(date.getTime()), contents);
        }
    }
    ParsedResult.maybeAppend(result.getNote(), contents);

    if (namesLength > 0) {
        Spannable styled = new SpannableString(contents.toString());
        styled.setSpan(new StyleSpan(android.graphics.Typeface.BOLD), 0, namesLength, 0);
        return styled;
    }
    return contents.toString();
}

private static Date parseDate(String s) {
    for (DateFormat currentFomat : DATE_FORMATS) {
        synchronized (currentFomat) {
            currentFomat.setLenient(false);
            Date result = currentFomat.parse(s, new ParsePosition(0));
            if (result != null) {
                return result;
            }
```

```
            }
        }
        return null;
    }

    @Override
    public int getDisplayTitle() {
        return R.string.result_address_book;
    }
}
```

18.4.2　日历处理

编写文件 CalendarResultHandler.java 加入处理日历信息，也就是处理 QR 码的日历项信息，具体实现代码如下所示。

```
public final class CalendarResultHandler extends ResultHandler {

    private static final DateFormat DATE_FORMAT = new SimpleDateFormat("yyyyMMdd", Locale.ENGLISH);
    private static final DateFormat DATE_TIME_FORMAT = new SimpleDateFormat("yyyyMMdd'T'HHmmss",
Locale.ENGLISH);

    public CalendarResultHandler(Activity activity, ParsedResult result) {
        super(activity, result);
    }

    @Override
    public CharSequence getDisplayContents() {
        CalendarParsedResult calResult = (CalendarParsedResult) getResult();
        StringBuilder result = new StringBuilder(100);
        ParsedResult.maybeAppend(calResult.getSummary(), result);
        Date start = calResult.getStart();
        String startString = start.toGMTString();
        appendTime(startString, result, false, false);

        Date end = calResult.getEnd();
        String endString = end.toGMTString();
        if (endString != null) {
            boolean sameStartEnd = startString.equals(endString);
            appendTime(endString, result, true, sameStartEnd);
        }

        ParsedResult.maybeAppend(calResult.getLocation(), result);
        ParsedResult.maybeAppend(calResult.getAttendees(), result);
        ParsedResult.maybeAppend(calResult.getDescription(), result);
        return result.toString();
    }

    private static void appendTime(String when, StringBuilder result, boolean end, boolean sameStartEnd) {
        if (when.length() == 8) {
            //只显示 year/month/day 格式日期
```

```
                Date date;
                synchronized (DATE_FORMAT) {
                    date = DATE_FORMAT.parse(when, new ParsePosition(0));
                }
                //如果是全天的，这是结束日期
                if (end && !sameStartEnd) {
                    date = new Date(date.getTime() - 24 * 60 * 60 * 1000);
                }
                ParsedResult.maybeAppend(DateFormat.getDateInstance().format(date.getTime()), result);
            } else {
                //本地时间
                Date date;
                synchronized (DATE_TIME_FORMAT) {
                    date = DATE_TIME_FORMAT.parse(when.substring(0, 15), new ParsePosition(0));
                }
                long milliseconds = date.getTime();
                if (when.length() == 16 && when.charAt(15) == 'Z') {
                    Calendar calendar = new GregorianCalendar();
                    int offset = calendar.get(Calendar.ZONE_OFFSET) + calendar.get(Calendar.DST_OFFSET);
                    milliseconds += offset;
                }
                ParsedResult.maybeAppend(DateFormat.getDateTimeInstance().format(milliseconds), result);
            }
        }

        @Override
        public int getDisplayTitle() {
            return R.string.result_calendar;
        }
    }
```

18.4.3 处理邮箱

编写文件 EmailAddressResultHandler.java，功能是将邮箱信息生成 QR 二维码，具体实现代码如下所示。

```
public final class EmailAddressResultHandler extends ResultHandler {

    public EmailAddressResultHandler(Activity activity, ParsedResult result) {
        super(activity, result);
    }

    @Override
    public CharSequence getDisplayContents() {
        EmailAddressParsedResult result = (EmailAddressParsedResult) getResult();
        StringBuilder contents = new StringBuilder(100);
        ParsedResult.maybeAppend(result.getEmailAddress(), contents);
        contents.trimToSize();
        return contents.toString();
    }
```

```
    @Override
    public int getDisplayTitle() {
        return R.string.result_email_address;
    }
}
```

到此为止，整个实例介绍完，有关 URL 网址、SMS 信息和 WiFi 二维码的生成过程，请读者参考本书附带光盘中的具体源码。本项目执行后的效果如图 18-1 所示。

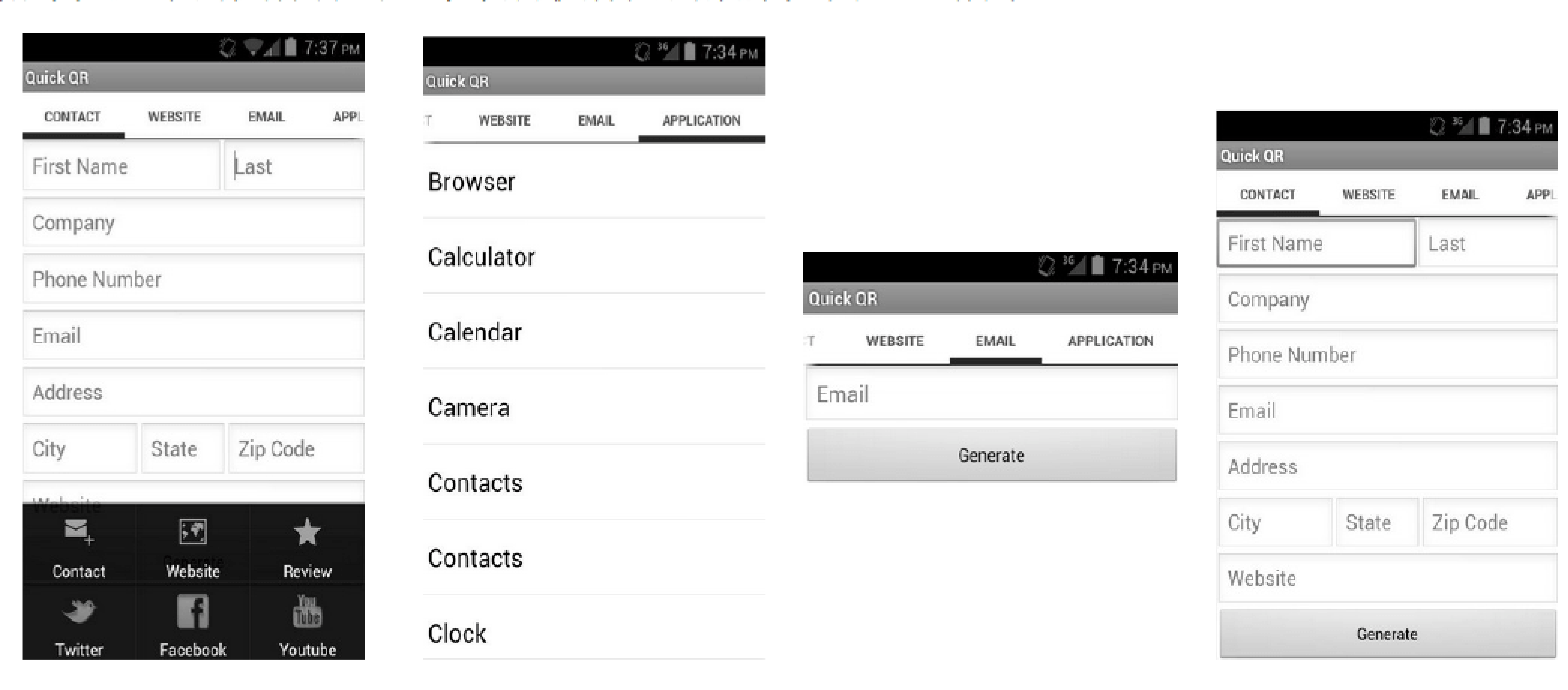


图 18-1　执行效果

第 19 章　骑行记录仪

随着国内外健身运动狂潮的兴起，纷纷诞生了很多健身运动俱乐部和健身组织，例如自驾俱乐部、登山俱乐部、跑吧、骑行俱乐部等。为了更好地了解自己的运动轨迹和行走路线，迫切需要开发一个能够记录行走或行车路线的 Android 应用程序。本章将通过一个综合实例的实现过程，详细讲解在 Android 设备中开发骑行记录仪的方法。读者可以以此实例为基础，开发其他类似的行车记录仪等应用程序。

19.1　选择线路规划目的地

知识点讲解：光盘:视频\知识点\第 19 章\选择线路规划目的地.avi

本 Android 应用程序实例提供了基于 GPS 线路计划和定位功能，支持从位置 A 到位置 B 路径规划，并且可以查看附近位置的单车停放处，并一步一步地指示行车路线生成路线图，可以实现卫星导航等功能。

实例	功能	源码路径
实例 19-1	骑行记录仪	光盘:\codes\19\BikeRoute

19.1.1　系统主 Activity 界面

本系统的主界面 Activity 是一个选择线路规划目的地的界面，Activity 实现文件是 Navigate.java，功能是获取用户输入的起始地址和目的地地址，根据地址实现路线规划功能。文件 Navigate.java 的具体实现代码如下所示。

```
public class Navigate extends Activity implements RouteListener {
	/**起始地址框 **/
	private transient AutoCompleteTextView startAddressField;
	/** 目的地地址框 **/
	private transient AutoCompleteTextView endAddressField;
	/** 停车管理 */
	private Parking prk;
	/**地址数据库 **/
	private AddressDatabase db;
	/** 运行搜索**/
	public boolean isSearching;
	protected AbstractContactAccessor mContactAccessor;
	private Intent searchIntent;
	protected BroadcastReceiver routeReceiver;
	protected BikeRouteApp app;
```

```
private RoutePlannerTask search;

/**计划对话框 **/
private static boolean mShownDialog;

/**路由 ID 产生器**/
Random random;

@Override
public final void onCreate(final Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
random = new Random();
app = ((BikeRouteApp) getApplication());
mContactAccessor = AbstractContactAccessor.getInstance();
requestWindowFeature(Window.FEATURE_RIGHT_ICON);
setContentView(R.layout.findplace);
setFeatureDrawableResource(Window.FEATURE_RIGHT_ICON, R.drawable.ic_bar_bikeroute);

searchIntent = new Intent();

isSearching = false;
mShownDialog = false;

//获取数据
db = app.getDb();
prk = new Parking(this);

//初始化域
startAddressField = (AutoCompleteTextView) findViewById(R.id.start_address_input);
endAddressField = (AutoCompleteTextView) findViewById(R.id.end_address_input);
final Button searchButton = (Button) findViewById(R.id.search_button);

//初始化适配器
final FindPlaceAdapter adapter = new FindPlaceAdapter(this,
            android.R.layout.simple_dropdown_item_1line);
startAddressField.setAdapter(adapter);
endAddressField.setAdapter(adapter);

//初始化搜索按钮
searchButton.setOnClickListener(new SearchClickListener());

/*自动填充的起始位置的反向地理编码*/

//手柄旋转
final Object[ ] data = (Object[ ]) getLastNonConfigurationInstance();
if (data != null) {
    isSearching = (Boolean) data[2];
    startAddressField.setText((String) data[3]);
    endAddressField.setText((String) data[4]);
    search = (RoutePlannerTask) data[5];
    if(search != null) {
        search.setListener(this);
```

```
            }
        }
    }

    public void onStart() {
        super.onStart();
        Thread t = new Thread() {
            public void run() {
                //初始化 geocoder
                final Geocoder geocoder = new Geocoder(Navigate.this);
                /*获取当前位置 */
                final LocationManager lm = (LocationManager)getSystemService(Context.LOCATION_
        SERVICE);

                Location self = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                if (self == null) {
                    self = lm.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                }
                if (self != null) {
                    try {
                        final Address startAddress = geocoder.getFromLocation(self.getLatitude(),
                                self.getLongitude(), 1).get(0);
                        startAddressField.setText(StringAddress.asString(startAddress));
                    } catch (Exception e) {
                        Log.e(e.getMessage(), "FindPlace - location: " + self);
                    }
                }
            }
        };
        t.run();
    }


    /**
     *从导航请求文本框处理启动地址地理编码
     *通过路线规划和行动，一旦计划完成后生成路线图
    */
    private class SearchClickListener implements OnClickListener {
        @Override
        public void onClick(final View view) {
            searchIntent.putExtra(RoutePlannerTask.PLAN_TYPE, RoutePlannerTask.ADDRESS_PLAN);
            searchIntent.putExtra(RoutePlannerTask.START_ADDRESS,
startAddressField.getText().toString());
            searchIntent.putExtra(RoutePlannerTask.END_ADDRESS,
endAddressField.getText().toString());
            requestRoute();
        }
    }

    /**
     *重写处理旋转跟踪显示对话框
     */
```

```
@Override
protected void onPrepareDialog(final int id, final Dialog dialog) {
        super.onPrepareDialog(id, dialog);
        if (id == R.id.plan) {
                mShownDialog = true;
        }
}

/**
 *要求根据规划服务的路线注册一个接收器来处理它
 */
private void requestRoute() {
        searchIntent.putExtra(RoutePlannerTask.ROUTE_ID, random.nextInt(2147483647));
        showDialog(R.id.plan);
        isSearching = true;
        search = new RoutePlannerTask(this, searchIntent);
        search.execute();
}

/**
*创建加载错误、警报对话框
*/
@Override
public Dialog onCreateDialog(final int id) {
        AlertDialog.Builder builder;
        ProgressDialog pDialog;
        Dialog dialog;
        switch(id) {
        case R.id.plan:
                pDialog = new ProgressDialog(this);
                pDialog.setCancelable(true);
                pDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
                pDialog.setMessage(getText(R.string.plan_msg));
                pDialog.setOnDismissListener(new OnDismissListener() {
                        @Override
                        public void onDismiss(final DialogInterface arg0) {
                                removeDialog(R.id.plan);
                                if (!Navigate.this.isSearching) {
                                        mShownDialog = false;
                                }
                        }
                });
                pDialog.setOnCancelListener(new OnCancelListener() {

                        @Override
                        public void onCancel(final DialogInterface arg0) {
                                        search.cancel(true);
                        }

                });
                dialog = pDialog;
```

```
                break;
            case R.id.plan_fail:
                builder = new AlertDialog.Builder(this);
                builder.setMessage(getText(R.string.planfail_msg)).setCancelable(
                        true).setPositiveButton(getString(R.string.ok),
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(final DialogInterface dialog,
                                    final int id) {
                            }
                        });
                dialog = builder.create();
                break;
            case R.id.ioerror:
                builder = new AlertDialog.Builder(this);
                builder.setMessage(getText(R.string.io_error_msg)).setCancelable(
                        true).setPositiveButton(getString(R.string.ok),
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(final DialogInterface dialog,
                                    final int id) {
                                dialog.dismiss();
                            }
                        });
                dialog = builder.create();
                break;
            case R.id.argerror:
                builder = new AlertDialog.Builder(this);
                builder.setMessage(getText(R.string.arg_error_msg)).setCancelable(
                        true).setPositiveButton(getString(R.string.ok),
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(final DialogInterface dialog,
                                    final int id) {
                                dialog.dismiss();
                            }
                        });
                dialog = builder.create();
                break;
            case R.id.reserror:
                builder = new AlertDialog.Builder(this);
                builder.setMessage(getText(R.string.result_error_msg)).setCancelable(
                        true).setPositiveButton(getString(R.string.ok),
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(final DialogInterface dialog,
                                    final int id) {
                                dialog.dismiss();
                            }
                        });
                dialog = builder.create();
                break;
```

```
        case R.id.about:
            builder = new AlertDialog.Builder(this);
            builder.setMessage(getText(R.string.about_message)).setCancelable(
                    true).setPositiveButton(getString(R.string.ok),
                    new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(final DialogInterface dialog,
                                final int id) {
                            dialog.dismiss();
                        }
                    });
            dialog = builder.create();
            break;
        default:
            dialog = null;
        }
        return dialog;
    }

    /**
     *创建选项菜单
     */
    @Override
    public final boolean onCreateOptionsMenu(final Menu menu) {
        final MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.navigate_menu, menu);
        return true;
    }

    @Override
    public final boolean onPrepareOptionsMenu(final Menu menu) {
        final MenuItem steps = menu.findItem(R.id.directions);
        final MenuItem back = menu.findItem(R.id.bike);
        final MenuItem stand    = menu.findItem(R.id.stand);
        steps.setVisible(false);
        if (app.getRoute() != null) {
            steps.setVisible(true);
        }
        back.setVisible(false);
        stand.setVisible(false);
        if (startAddressField.getText().length() > 0) {
            stand.setVisible(true);
            if (prk.isParked()) {
                back.setVisible(true);
            }
        }
        return super.onPrepareOptionsMenu(menu);
    }

    /**
     * 根据用户选择处理事件程序
     * @return true if option selected.
     */
```

```
    @Override
    public boolean onOptionsItemSelected(final MenuItem item) {
        Intent intent;
        switch(item.getItemId()) {
        case R.id.prefs:
            intent = new Intent(this, Preferences.class);
            startActivity(intent);
            break;
        case R.id.directions:
            intent = new Intent(this, DirectionsView.class);
            startActivity(intent);
            break;
        case R.id.map:
            intent = new Intent(this, LiveRouteMap.class);
            startActivity(intent);
            break;
        case R.id.bike:
            searchIntent.putExtra(RoutePlannerTask.PLAN_TYPE, RoutePlannerTask.BIKE_PLAN);
            searchIntent.putExtra(RoutePlannerTask.START_ADDRESS,
startAddressField.getText().toString());
            requestRoute();
            break;
        case R.id.stand:
            searchIntent.putExtra(RoutePlannerTask.PLAN_TYPE, RoutePlannerTask.STANDS_PLAN);
            searchIntent.putExtra(RoutePlannerTask.START_ADDRESS,
startAddressField.getText().toString());
            requestRoute();
            break;
        case R.id.about:
            showDialog(R.id.about);
            break;
        case R.id.contacts:
            startActivityForResult(mContactAccessor.getPickContactIntent(), 0);
        }
        return true;
    }

    /**
     * 根据搜索任务处理回调函数
     */

    @Override
    public void searchComplete(final Integer msg, final Route route) {
        if (msg != null) {
            try {
                dismissDialog(R.id.plan);
            } catch (Exception e) {
                Log.e("Navigate", e.getMessage());
            }
            if (msg == R.id.result_ok) {
                db.insert(startAddressField.getText().toString());
```

```
                if (!"".equals(endAddressField.getText().toString())) {
                    db.insert(endAddressField.getText().toString());
                }
                final Intent map = new Intent(this, LiveRouteMap.class);
                app.setRoute(route);
                startActivity(map);
            } else {
                showDialog(msg);
            }
        }
    }

    /**
     *回调处理用户选择的地址
     */
    @Override
    protected void onActivityResult(final int requestCode, final int resultCode, final Intent data) {
        if (resultCode == RESULT_OK) {
            loadContactAddress(data.getData());
        }
    }

    /**
     *在后台载入存储地址，并设置为目的地
     */
    private void loadContactAddress(final Uri contactUri) {
        final AsyncTask<Uri, Void, String> task = new AsyncTask<Uri, Void, String>() {

            @Override
            protected String doInBackground(Uri... uris) {
                return mContactAccessor.loadAddress(getContentResolver(), uris[0]);
            }

            @Override
            protected void onPostExecute(final String result) {
              endAddressField.setText(result);
            }
        };

        task.execute(contactUri);
    }

    /**
     *在旋转的对话与文本框中重写保存的搜索任务
     */
    @Override
    public Object onRetainNonConfigurationInstance() {
        Object[ ] objs = new Object[6];
        objs[1] = mShownDialog;
        objs[2] = isSearching;
        objs[3] = startAddressField.getText().toString();
        objs[4] = endAddressField.getText().toString();
```

```
            objs[5] = search;
            return objs;
        }
        @Override
        public void searchCancelled() {
            isSearching = false;
            search = null;
        }
        @Override
        public Context getContext() {
            return this;
        }
}
```

19.1.2　布局文件 capture.xml

文件 Navigate.java 调用的布局文件是 capture.xml，功能是为用户提供输入始发地和目的地地点的表单。文件 findplace.xml 的具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:focusableInTouchMode="true"
android:focusable="true"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:gravity="center"
android:paddingLeft="5px"
android:paddingTop="5px"
android:paddingRight="5px">
<com.nanosheep.bikeroute.view.StepView
android:layout_width="fill_parent"
android:id="@+id/search"
android:layout_height="wrap_content"
android:orientation="vertical"
android:gravity="center"
android:layout_alignParentTop="true"
android:padding="10px">
<TextView
android:id="@+id/start_address_view"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/start"
android:gravity="center_vertical">
</TextView>
<AutoCompleteTextView
android:id="@+id/start_address_input"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="@string/start_input"
android:textSize="18sp"
```

```
android:gravity="center"
android:layout_below="@id/start_address_view">
</AutoCompleteTextView>
<TextView
android:id="@+id/end_address_view"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/end"
android:gravity="center_vertical"
android:layout_below="@id/start_address_input">
</TextView>
<AutoCompleteTextView
android:id="@+id/end_address_input"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="@string/end_input"
android:textSize="18sp"
android:gravity="center"
android:layout_below="@id/end_address_view">
</AutoCompleteTextView>
<Button
android:id="@+id/search_button"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/search"
android:gravity="center"

android:layout_below="@id/end_address_input">
</Button>
</com.nanosheep.bikeroute.view.StepView>
</LinearLayout>
```

主界面的执行效果如图 19-1 所示。

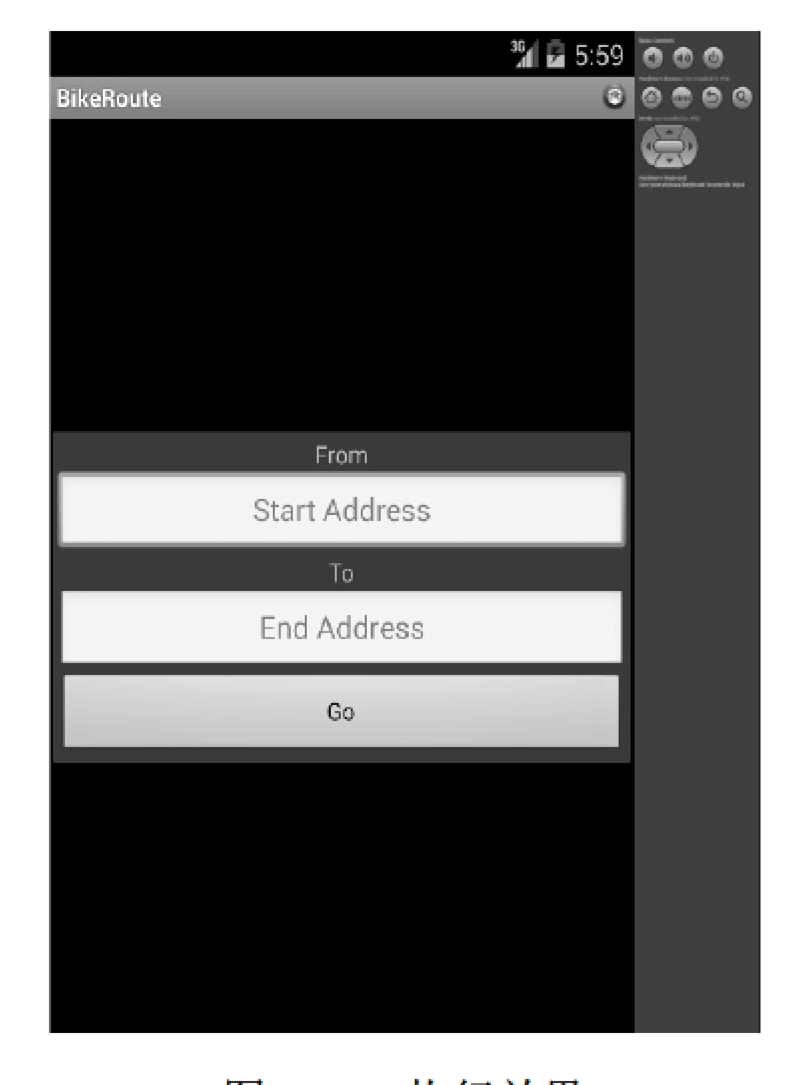


图 19-1　执行效果

19.2 Adapter 适配器处理

知识点讲解：光盘:视频\知识点\第 19 章\Adapter 适配器处理.avi

在 Android 系统中，Adapter 是连接后端数据和前端显示的适配器接口，是数据和 UI（View）之间一个重要的纽带，在常见的 View（List View,Grid View）等地方都需要用到 Adapter。在本实例中，需要在 View 视图中显示后台地址数据。

文件 DirectionListAdapter.java 实现位置方向适配器处理，具体实现代码如下所示。

```
/**
 *方向适配器对象
*/
public class DirectionListAdapter extends ArrayAdapter&lt;Segment&gt; {
	/**  布局 inflater **/
	private final transient LayoutInflater inflater;
	/**  单元  **/
	private String unit;
	public DirectionListAdapter(final Context context, final int textView) {
		super(context, textView);
		inflater = LayoutInflater.from(context);
		this.populate();
	}

	/**
	 *将当前段列表填充到适配器，清除任何已经存在的数据
	 */
	public void populate() {
		clear();
		final SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(getContext());
		unit = settings.getString("unitsPref", "km");
		BikeRouteApp app = (BikeRouteApp) getContext().getApplicationContext();
		for (Segment s : app.getRoute().getSegments()) {
			add(s);
		}
	}

	/**
	 *得到一个路径地名显示视图
	 */

	@Override
	public View getView(final int position, final View convertView, final ViewGroup parent) {
		final Segment segment = getItem(position);
		final View view = inflater.inflate(R.layout.direction_item, null);
		final TextView turn = (TextView) view.findViewById(R.id.turn);
		final TextView distance = (TextView) view.findViewById(R.id.distance);
		final TextView length = (TextView) view.findViewById(R.id.length);
```

```
            turn.setText(segment.getInstruction());
            if ("km".equals(unit)) {
                length.setText(Convert.asMeterString(segment.getLength()));
                distance.setText("(" + Convert.asKilometerString(segment.getLength()) + ")");

            } else {
                length.setText(Convert.asFeetString(segment.getLength()));
                distance.setText("(" + Convert.asMilesString(segment.getLength()) + ")");
            }

            return view;
    }
}
```

文件 FindPlaceAdapter.java 是寻找位置适配器文件，功能是检索系统中所需要的所有的位置信息，具体实现代码如下所示。

```
public class FindPlaceAdapter extends ArrayAdapter<String> {
    private final Geocoder geocoder;
    /**以前的地址数据库 **/
    private final AddressDatabase db;

    public FindPlaceAdapter(final Context context, final int resource,
            final int txtViewResId) {
        super(context, resource, txtViewResId);
        geocoder = new Geocoder(context);
        db = ((BikeRouteApp) context.getApplicationContext()).getDb();
    }

    public FindPlaceAdapter(final Context context, final int resource) {
        super(context, resource);
        geocoder = new Geocoder(context);
        db = ((BikeRouteApp) context.getApplicationContext()).getDb();
    }

    /**
     *从地理编码服务检索数据以取代 ArrayAdapter 过滤器
     * @return a Filter object
     */

    @Override
    public Filter getFilter() {
        return new GeoFilter();
    }

    /**
     *伪滤波器，返回基于输入的字符序列的地址
     */

    private class GeoFilter extends Filter {
        private List<Address> addresses;
```

```
        public GeoFilter() {
            super();
            addresses = new ArrayList<Address>();
        }

        /**
         *执行一个 SQL 语句，在 SQL 语句中使用字符序列作为一个地理编码，过滤掉系统中保存的以前的检
索码信息 */
        @Override
        protected Filter.FilterResults performFiltering(final CharSequence ch) {
            final Filter.FilterResults res = new Filter.FilterResults();
            final Set<String> results = new HashSet<String>();
            if (ch == null) {
                res.count = 0;
            } else {
                final String addressInput = ch.toString();

                //Add results from db
                results.addAll(db.selectLike(ch));
                //Search using geocoder
                try {
                    addresses = geocoder.getFromLocationName(addressInput, 5);
                    for (Address address : addresses) {
                        results.add(StringAddress.asString(address));
                    }

                    res.count = results.size();
                    res.values = results;
                } catch (IOException e) {
                    res.count = -1; //pass result back to ui thread to show message
                }

            }
            return res;
        }

        /**
         * 显示结果
         */
        @SuppressWarnings("unchecked")
        @Override
        protected void publishResults(final CharSequence constraint,
                final FilterResults results) {
            if (results.count > 0) {
                clear();
                for (String address : (Set<String>) results.values) {
                    add(address);
                }
                FindPlaceAdapter.this.notifyDataSetChanged();
            } else if (results.count == -1) {
                //Show an io error message if an exception was thrown
```

```
                    //((Activity) getContext()).showDialog(R.id.ioerror);
                    FindPlaceAdapter.this.notifyDataSetInvalidated();
                } else {
                    FindPlaceAdapter.this.notifyDataSetInvalidated();
                }
            }
        }
}
```

19.3　生成路线图

知识点讲解：光盘:视频\知识点\第 19 章\生成路线图.avi

本实例的核心功能是生成需要的行驶路线图，此功能需要调用 Android 系统的地图定位功能，并且还需要借助本系统的导航服务功能。在本节的内容中，将详细讲解生成路线图功能的实现过程。

19.3.1　实时导航服务

文件 NavigationService.java 的功能是提供实时的导航服务，并使用 GPS 和通知更新反映路径中的当前位置。文件 NavigationService.java 的具体实现代码如下所示。

```
public class NavigationService extends Service implements LocationListener{
    private final IBinder mBinder = new LocalBinder();
     private NotificationManager mNM;
     private LocationManager mLocationManager;
     private BikeRouteApp app;
     private Notification notification;
     private PendingIntent contentIntent;
    public class LocalBinder extends Binder {
        public NavigationService getService() {
            return NavigationService.this;
        }
    }

     /* (non-Javadoc)
      * @see android.app.Service#onBind(android.content.Intent)
      */
     @Override
     public IBinder onBind(Intent arg0) {
          return mBinder;
     }

     @Override
    public void onCreate() {
        mNM = (NotificationManager)getSystemService(NOTIFICATION_SERVICE);
        /* Get location manager. */
          mLocationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
        app = (BikeRouteApp) getApplication();
        int icon = R.drawable.bikeroute;
```

```
        CharSequence tickerText = "";
        long when = System.currentTimeMillis();

        notification = new Notification(icon, tickerText, when);
        notification.flags |= Notification.FLAG_ONGOING_EVENT;
        Intent notificationIntent = new Intent(this, LiveRouteMap.class);
        notificationIntent.putExtra(getString(R.string.jump_intent), true);
        contentIntent = PendingIntent.getActivity(this, 0, notificationIntent, 0);
        mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
        if (app.getRoute() != null) {
          notification.setLatestEventInfo(app, getText(R.string.notify_title),
                      app.getSegment().getInstruction(), contentIntent);
          mNM.notify(R.id.notifier, notification);
        }
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.i("LocalService", "Received start id " + startId + ": " + intent);
        // We want this service to continue running until it is explicitly
        // stopped, so return sticky
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
     mLocationManager.removeUpdates(this);
     mNM.cancelAll();
    }

    /**
     * 监听位置更新，并及时反馈到路线图中，然后在更新状态栏中发布提醒信息
    */
    @Override
     public void onLocationChanged(Location location) {
     if (app.getRoute() != null) {
          Intent update = new Intent((String) getText(R.string.navigation_intent));
          //找到最近的点
          GeoPoint self = new GeoPoint(location);
          List<GeoPoint> near = app.getRoute().nearest(self, 2);

          double range = range(self, near.get(0), near.get(1)) - 50;
          double accuracy = location.getAccuracy();
          if (range > accuracy) {
               update.putExtra((String) getText(R.string.replan), true);
               String logMsg = "Range=" + range + ",Self="+self+",near points:" + near + ",near points dist=" +
near.get(0).distanceTo(near.get(1)) + ",accuracy=" + accuracy;
               Log.e("Replanned", logMsg);
               notification.setLatestEventInfo(app, getText(R.string.notify_title),
                         getText(R.string.replanning), contentIntent);
          } else if (near.get(0).equals(app.getRoute().getEndPoint())) { //If we've arrived, shutdown and signal.
```

```
                update.putExtra((String) getText(R.string.arrived), true);
                notification.setLatestEventInfo(app, getText(R.string.notify_title),
                        getText(R.string.arrived), contentIntent);
            }   else {
                update.putExtra((String) getText(R.string.point), near.get(0));
                app.setSegment(app.getRoute().getSegment(near.get(0)));
                notification.setLatestEventInfo(app, getText(R.string.notify_title),
                        app.getSegment().getInstruction(), contentIntent);
            }
            sendBroadcast(update);

            mNM.notify(R.id.notifier, notification);
    }
    }

    /* (non-Javadoc)
     * @see android.location.LocationListener#onProviderDisabled(java.lang.String)
     */
    @Override
    public void onProviderDisabled(String provider) {
        // TODO Auto-generated method stub

    }

    /* (non-Javadoc)
     * @see android.location.LocationListener#onProviderEnabled(java.lang.String)
     */
    @Override
    public void onProviderEnabled(String provider) {
        // TODO Auto-generated method stub

    }

    /* (non-Javadoc)
     * @see android.location.LocationListener#onStatusChanged(java.lang.String, int, android.os.Bundle)
     */
    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        // TODO Auto-generated method stub

    }

    /**
     *从路径 P1 -> P2 获取交叉轨道误差的 P0
     */

    private double range(final GeoPoint p0, final GeoPoint p1, final GeoPoint p2) {
        double dist = Math.asin(Math.sin(p1.distanceTo(p0)/BikeRouteConsts.EARTH_RADIUS) *
                    Math.sin(p1.bearingTo(p0) - p1.bearingTo(p2))) *
                    BikeRouteConsts.EARTH_RADIUS;
```

```
            return Math.abs(dist);
        }
}
```

19.3.2 线路计划监听服务

文件 RouteListener.java 实现线路位置计划监听服务，具体实现代码如下所示。

```
public interface RouteListener {
        public void searchComplete(Integer msg, Route route);
        public void searchCancelled();
        public Context getContext();
}
```

19.3.3 线路任务服务

文件 RoutePlannerTask.java 的功能是显示搜索任务和计划对话框，并将结果过渡到一个地图中来显示路径。文件 RoutePlannerTask.java 的具体实现代码如下所示。

```
public class RoutePlannerTask extends AsyncTask<Void, Void, Integer> {
        /** Route planner service consts. **/
        public static final String PLAN_TYPE = "plan_type";
        public static final int BIKE_PLAN = 0;
        public static final int GEOPOINT_PLAN = 1;
        public static final int REPLAN_PLAN = 2;
        public static final int STANDS_PLAN = 3;
        public static final int ADDRESS_PLAN = 4;
        public static final String START_ADDRESS = "start_address";
        public static final String END_ADDRESS = "end_address";
        public static final String START_LOCATION = "start_location";
        public static final String END_POINT = "end_point";
        public static final String ROUTE_ID = "route_id";
        private RouteManager planner;
       protected String startAddressInput;
       protected String endAddressInput;
        private RouteListener mAct;
        private Intent mIntent;
             public RoutePlannerTask(RouteListener act, Intent intent) {
                     super();
                     mIntent = intent;
                     mAct = act;
                     planner = new RouteManager(mAct.getContext());
             }
             public void setListener(final RouteListener listener) {
               mAct = listener;
             }
             @Override
             protected void onPreExecute() {
             }
             @Override
             protected Integer doInBackground(Void... arg0) {
```

```
int msg = R.id.plan_fail;
planner.setRouteId(mIntent.getIntExtra(ROUTE_ID, 0));
final String startAddressInput = mIntent.getStringExtra(START_ADDRESS);
final String endAddressInput = mIntent.getStringExtra(END_ADDRESS);
        switch(mIntent.getIntExtra(PLAN_TYPE, ADDRESS_PLAN)) {
    case ADDRESS_PLAN:
        if ("".equals(startAddressInput) || "".equals(endAddressInput)) {
            msg = R.id.argerror;
        } else {
            msg = R.id.result_ok;
            try {
                planner.setStart(startAddressInput);
                planner.setDest(endAddressInput);
            } catch (Exception e) {
                msg = R.id.ioerror;
            }
        }
        break;
    case BIKE_PLAN:
        final Parking prk = new Parking(mAct.getContext());
        if ("".equals(startAddressInput)) {
            msg = R.id.argerror;
        } else {
            try {
                planner.setStart(startAddressInput);
                planner.setDest(prk.getLocation());
            } catch (Exception e) {
                msg = R.id.ioerror;
            }
        }
        break;
    case STANDS_PLAN:
        if ("".equals(startAddressInput)) {
            msg = R.id.argerror;
        } else {
            msg = R.id.result_ok;
            try {
                planner.setStart(startAddressInput);
                planner.setDest(Stands.getNearest(planner.getStart(), mAct.getContext()));
            } catch (Exception e) {
                msg = R.id.ioerror;
            }
        }
        break;
    case REPLAN_PLAN:
        final Location start = mIntent.getParcelableExtra(START_LOCATION);
        final GeoPoint dest = mIntent.getParcelableExtra(END_POINT);
        msg = R.id.result_ok;
        planner.setStart(start);
        planner.setDest(dest);
        break;
```

```
                default:
                    msg = R.id.plan_fail;
                }
                  try {
                    if ((msg == R.id.result_ok) && !planner.showRoute()) {
                        msg = R.id.plan_fail;
                    }
                } catch (Exception e) {
                    msg = R.id.ioerror;
                }
                return msg;
        }
        @Override
        protected void onPostExecute(final Integer msg) {
          mAct.searchComplete(msg,    planner.getRoute());
        }

        @Override
        protected void onCancelled() {
          mAct.searchCancelled();
          super.onCancelled();
        }
}
```

19.3.4　在地图中显示行驶线路

文件 LiveRouteMap.java 的功能是调用 19.3.1 节到 19.3.3 节中的 3 个导航服务，根据用户输入的起始地址和目标地址，在地图中显示对应的线路路径。文件 LiveRouteMap.java 的具体实现代码如下所示。

```
public class LiveRouteMap extends SpeechRouteMap implements RouteListener {
    protected Intent searchIntent;
    protected boolean mShownDialog;
    private RoutePlannerTask search;
    private boolean liveNavigation;
    private Segment lastSegment;
    private boolean spoken;
    private boolean arrived;
    private NavigationService mBoundService;
    private NavigationReceiver mBroadcastReceiver = new NavigationReceiver();
    private ServiceConnection mConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className, IBinder service) {
            mBoundService = ((NavigationService.LocalBinder)service).getService();
        }

        public void onServiceDisconnected(ComponentName className) {
            mBoundService = null;
        }
    };
    private boolean mIsBound;

    @Override
```

```
public void onCreate(final Bundle savedState) {
        super.onCreate(savedState);
        //Handle rotations
        final Object[ ] data = (Object[ ]) getLastNonConfigurationInstance();
        arrived = false;
        if (data != null) {
                isSearching = (Boolean) data[2];
                search = (RoutePlannerTask) data[3];
                if(search != null) {
                        search.setListener(this);
                }
                spoken = (Boolean) data[4];
                arrived = (Boolean) data[1];
        }
        registerReceiver(mBroadcastReceiver,
                        new IntentFilter(getString(R.string.navigation_intent)));
}

@Override
public Object onRetainNonConfigurationInstance() {
        Object[ ] objs = new Object[5];
        objs[0] = directionsVisible;
        objs[1] = arrived;
        objs[2] = isSearching;
        objs[3] = search;
        objs[4] = spoken;
       return objs;
}

@Override
public final boolean onPrepareOptionsMenu(final Menu menu) {
        final MenuItem replan = menu.findItem(R.id.replan);
        final MenuItem stopService = menu.findItem(R.id.stop_nav);
        if (app.getRoute() != null) {
                replan.setVisible(true);
        }
        if (mIsBound) {
                stopService.setVisible(true);
        }
        return super.onPrepareOptionsMenu(menu);
}

@Override
public void showStep() {
        super.showStep();
        if(mSettings.getBoolean("gps", false)) {
                mLocationOverlay.followLocation(true);
        } else {
                mLocationOverlay.followLocation(false);
        }
}
```

```
@Override
public void hideStep() {
    super.hideStep();
    mLocationOverlay.followLocation(false);
}
private void replan() {
    isSearching = true;
    try {
        dismissDialog(R.id.plan_fail);
    } catch (Exception e) {
        Log.e("Replanner", "Fail dialog not shown!");
    }
    showDialog(R.id.plan);

    Location self = mLocationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

    if (self == null) {
        self = mLocationOverlay.getLastFix();
    }
    if (self == null) {
        self = mLocationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
    }
    if (self != null) {
        searchIntent = new Intent();
        searchIntent.putExtra(RoutePlannerTask.ROUTE_ID, app.getRoute().getRouteId());
        searchIntent.putExtra(RoutePlannerTask.PLAN_TYPE, RoutePlannerTask.REPLAN_PLAN);
        searchIntent.putExtra(RoutePlannerTask.START_LOCATION, self);
        searchIntent.putExtra(RoutePlannerTask.END_POINT,
                app.getRoute().getPoints().get(app.getRoute().getPoints().size() - 1));
        LiveRouteMap.this.search = new RoutePlannerTask(LiveRouteMap.this, searchIntent);
        LiveRouteMap.this.search.execute();
    } else {
        dismissDialog(R.id.plan);
        showDialog(R.id.plan_fail);
    }

}


@Override
public Dialog onCreateDialog(final int id) {
    Dialog dialog;
    AlertDialog.Builder builder;
    switch(id) {
    case R.id.gps:
        builder = new AlertDialog.Builder(this);
        builder.setMessage(R.string.gps_msg);
        builder.setCancelable(false);
        builder.setPositiveButton(getString(R.string.ok),
            new DialogInterface.OnClickListener() {
```

```
                    @Override
                    public void onClick(final DialogInterface dialog,
                            final int id) {
                        showGpsOptions();
                    }
                });
            builder.setTitle(R.string.gps_msg_title);
            dialog = builder.create();
            break;
        case R.id.plan:
            ProgressDialog pDialog = new ProgressDialog(this);
            pDialog.setCancelable(true);
            pDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            pDialog.setMessage(getText(R.string.plan_msg));
            pDialog.setOnDismissListener(new OnDismissListener() {
                @Override
                public void onDismiss(final DialogInterface arg0) {
                    removeDialog(R.id.plan);
                }
                });
            pDialog.setOnCancelListener(new OnCancelListener() {

                @Override
                public void onCancel(final DialogInterface arg0) {
                    if (search != null) {
                        search.cancel(true);
                    }
                    isSearching = false;
                }

            });
            dialog = pDialog;
            break;
        case R.id.plan_fail:
            builder = new AlertDialog.Builder(this);
            builder.setMessage(R.string.planfail_msg);
            builder.setCancelable(
                true).setPositiveButton(getString(R.string.ok),
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(final DialogInterface dialog,
                            final int id) {
                    }
                });
            dialog = builder.create();
            break;
        default:
            dialog = super.onCreateDialog(id);
        }
        return dialog;
    }
```

```
@Override
public boolean onOptionsItemSelected(final MenuItem item) {
    switch (item.getItemId()) {
    case R.id.replan:
        if (!mIsBound) {
            doBindService();
        }
        replan();
        break;
    case R.id.stop_nav:
        doUnbindService();
        this.finish();
        break;
    case R.id.turnbyturn:
        spoken = true;
    default:
        return super.onOptionsItemSelected(item);
    }
    return true;
}
private void doBindService() {
    bindService(new Intent(LiveRouteMap.this,
            NavigationService.class), mConnection, Context.BIND_AUTO_CREATE);
    mIsBound = true;
}

private void doUnbindService() {
    if (mIsBound) {
        // Detach our existing connection
        unbindService(mConnection);
        mIsBound = false;
    }
}
@Override
public void onDestroy() {
    super.onDestroy();
    doUnbindService();
    unregisterReceiver(mBroadcastReceiver);
}


@Override
public void onStart() {
    super.onStart();
    liveNavigation = mSettings.getBoolean("gps", false);
    if (app.getRoute() != null) {
        //Disable live navigation for non GB routes to comply with Google tos
        liveNavigation = !"GB".equals(app.getRoute().getCountry()) ? false : liveNavigation;
        if (tts && directionsVisible && !isSearching) {
            speak(app.getSegment());
            lastSegment = app.getSegment();
```

```
            }
            if (liveNavigation) {
                if(!mLocationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
                    showDialog(R.id.gps);
                }
                doBindService();
            } else {
                doUnbindService();
            }
        }
    }



    private class NavigationReceiver extends BroadcastReceiver {

        /* (non-Javadoc)
         * @see android.content.BroadcastReceiver#onReceive(android.content.Context, android.content.Intent)
         */
        @Override
        public void onReceive(Context context, Intent intent) {
            if (liveNavigation && directionsVisible && !arrived && !isSearching) {
                if (intent.getBooleanExtra(getString(R.string.replan), false)) {
                    isSearching = true;
                    replan();
                } else if (intent.getBooleanExtra(getString(R.string.arrived), false)) {
                    arrive();
                    spoken = true;
                } else {
                    GeoPoint current = (GeoPoint) intent.getExtras().get(getString(R.string.point));
                    if (!app.getSegment().equals(lastSegment)) {
                        lastSegment = app.getSegment();
                        spoken = false;
                    }

                    //Get next point
                    ListIterator<GeoPoint> it = app.getRoute().getPoints().listIterator(
                            app.getRoute().getPoints().indexOf(current) + 1);
                    GeoPoint next = it.hasNext() ? it.next() : current;

                    //Speak directions if the next point is a new segment
                    //and have not spoken already
                    if (!spoken && !app.getSegment().equals(app.getRoute().getSegment(next)) && tts) {
                            speak(app.getRoute().getSegment(next));
                            spoken = true;
                    }
                    showStep();
                    traverse(current);
                }
            }
        }
```

```
    }

    @Override
    public void searchComplete(Integer msg, Route route) {
        try {
            dismissDialog(R.id.plan);
        } catch (Exception e)   {

        }

        isSearching = false;
        if (msg != null) {

            if (msg == R.id.result_ok) {
                app.setRoute(route);
                app.setSegment(app.getRoute().getSegments().get(0));
                mOsmv.getController().setCenter(app.getSegment().startPoint());
                traverse(app.getSegment().startPoint());
                arrived = false;
                if (directionsVisible) {
                    showStep();
                    if (tts) {
                        speak(app.getSegment());
                        spoken = true;
                    }
                }
            } else {
                showDialog(msg);
            }
        }
    }

    @Override
    public void searchCancelled() {
        isSearching = false;
        search = null;
    }

    private void arrive() {
        doUnbindService();
        arrived = true;
        app.setSegment(app.getRoute().getSegment(app.getRoute().getEndPoint()));
        traverse(app.getRoute().getEndPoint());
        showStep();
        if (tts) {
            directionsTts.speak(getString(R.string.arrived_speech), TextToSpeech.QUEUE_ADD, null);
        }
    }

    private void showGpsOptions() {
      startActivity(new Intent(
```

```
                    android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
        }
}
```

19.3.5　生成导航视图

文件 DirectionsView.java 的功能是，根据用户输入的地址参数生成导航视图，能够根据用户的选择显示不同的视图界面，例如地图界面、关于我们、海拔、首选设置项等。文件 DirectionsView.java 的具体实现代码如下所示。

```
public class DirectionsView extends ListActivity {
        /** Route object. **/
        private Route route;
        /** Units. **/
        private String unit;
        private TextView header;
        private TextView footer;

        @Override
        public void onCreate(final Bundle in) {
                requestWindowFeature(Window.FEATURE_RIGHT_ICON);
                setFeatureDrawableResource(Window.FEATURE_RIGHT_ICON, R.drawable.ic_bar_bikeroute);
                super.onCreate(in);
                //Create a header to display route distance
                header = new TextView(this);
                getListView().addHeaderView(header, "", false);
                //Create a footer to display warnings & copyrights
                footer = new TextView(this);
                getListView().addFooterView(footer, "", false);
                //Add the list of directions and set it filterable
                setListAdapter(new DirectionListAdapter(this, R.layout.direction_item));
                getListView().setTextFilterEnabled(true);
        }

        @Override
        public void onStart() {
                super.onStart();
                final SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(this);
                unit = settings.getString(getString(R.string.prefs_units), getString(R.string.km));

                route = ((BikeRouteApp)getApplication()).getRoute();

                setTitle(route.getName());

                //Create a header for the list
                StringBuffer sBuf = new StringBuffer("Total distance: ");
                if (getString(R.string.km).equals(unit)) {
                        sBuf.append(Convert.asMeterString(route.getLength()));
                        sBuf.append(" (");
                        sBuf.append(Convert.asKilometerString(route.getLength()));
```

```
                sBuf.append(')');
        } else {
                sBuf.append(Convert.asFeetString(route.getLength()));
                sBuf.append(" (");
                sBuf.append(Convert.asMilesString(route.getLength()));
                sBuf.append(')');
        }
        header.setText(sBuf.toString());

        sBuf = new StringBuffer();
        if (route.getWarning() != null) {
                sBuf.append(route.getWarning());
                sBuf.append('\n');
        }
        if (route.getCopyright() != null) {
                sBuf.append(route.getCopyright());
        }
        footer.setText(sBuf.toString());
        ((DirectionListAdapter) getListAdapter()).populate();
}

@Override
protected void onListItemClick(final ListView l, final View v,
                final int position, final long id) {
        ((BikeRouteApp)getApplication()).
        setSegment(route.getSegments().get(position -1));
        final Intent intent = new Intent(this, LiveRouteMap.class);

        intent.putExtra(getString(R.string.jump_intent), true);
        startActivity(intent);
}

/**
 * 创建选项菜单
 * @return true if menu created.
 */
@Override
public final boolean onCreateOptionsMenu(final Menu menu) {
        final MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.directions_menu, menu);
        return true;
}

/**
 *根据用户的选择来到不同的视图界面
 */
@Override
public boolean onOptionsItemSelected(final MenuItem item) {
        Intent intentDir = null;
        switch (item.getItemId()) {
        case R.id.navigate:
```

```
                intentDir = new Intent(this, Navigate.class);
                break;
            case R.id.map:
                intentDir = new Intent(this, LiveRouteMap.class);
                break;
            case R.id.prefs:
                intentDir = new Intent(this, Preferences.class);
                break;
            case R.id.about:
                showDialog(R.id.about);
                return true;
            case R.id.elevation:
                XYMultipleSeriesDataset elevation = route.getElevations();
                final XYMultipleSeriesRenderer renderer = route.getChartRenderer();
                if (!getString(R.string.km).equals(unit)) {
                    elevation = Convert.asImperial(elevation);
                    renderer.setYTitle(getString(R.string.ft));
                   renderer.setXTitle(getString(R.string.meters));
                }
                renderer.setYAxisMax(elevation.getSeriesAt(0).getMaxY() + 200);
                intentDir = ChartFactory.getLineChartIntent(this, elevation, renderer);
            }
            startActivity(intentDir);
            return true;
        }

        /**
         *创建对话框界面
         */

        @Override
        public Dialog onCreateDialog(final int id) {
            final AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setMessage(getText(R.string.about_message)).setCancelable(
                    true).setPositiveButton(getString(R.string.ok),
                    new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(final DialogInterface dialog,
                                final int id) {
                            dialog.dismiss();
                        }
                    });
            return builder.create();
        }

}
```

导航视图界面对应的布局文件是 direction_item.xml，具体实现代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
```

```
android:background="#ffffffff"
android:padding="6dip"
android:id="@+id/directions_item">
<TextView
 android:id="@+id/turn"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:textSize="18px"
     android:textColor="#000000"
     android:text="Road and turn." />
<TextView
     android:id="@+id/length"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:textSize="10px"
     android:textColor="#5E72F7"
     android:text="Length of step (total distance in km)"
     android:layout_below="@id/turn" />
<TextView
     android:id="@+id/distance"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:textSize="10px"
     android:textColor="#49D611"
     android:paddingLeft="5px"
     android:text="Length of step (total distance in km)"
     android:layout_toRightOf="@id/length"
     android:layout_below="@id/turn" />

</RelativeLayout>
```

路径导航视图界面的执行效果如图 19-2 所示。

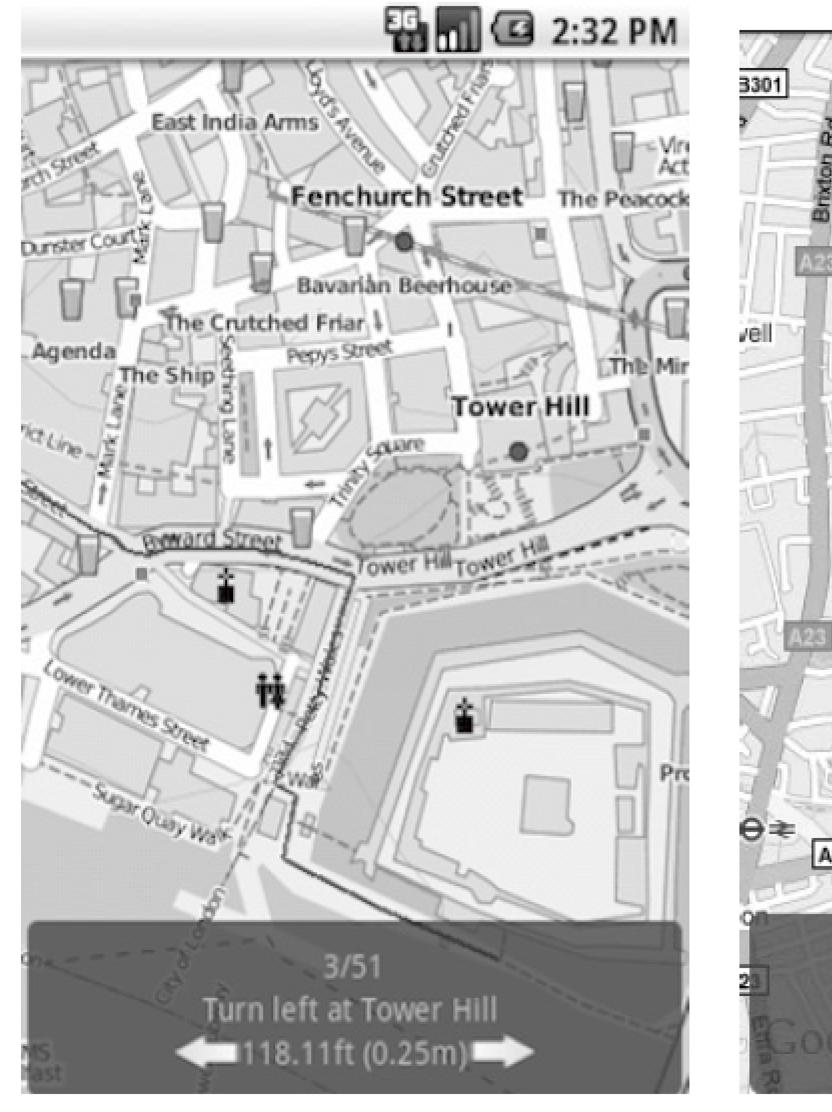




图 19-2　路径导航视图界面

19.4 街道分析

知识点讲解：光盘:视频\知识点\第 19 章\街道分析.avi

为了实现精准的线路规划和定位，本实例借助 http://www.cyclestreets.net/提供的街道数据，这样在规划线路时可以为我们构建更加精准的骑行线路。街道分析功能的实现文件是 CycleStreetsParser.java，功能是建立和 CycleStreetsParser API 的连接，构建 XML 街道数据。文件 CycleStreetsParser.java 的具体实现代码如下所示。

```
public class CycleStreetsParser extends XMLParser implements Parser {
	/** Distance covered. **/
	private double distance;

	public CycleStreetsParser(final String feedUrl) {
		super(feedUrl);
	}

	public final Route parse() {
		final Segment segment = new Segment();
		final Route route = new Route();
		route.setCopyright("Route planning by CycleStreets.net");
		final RootElement root = new RootElement(MARKERS);
		final Element marker = root.getChild(MARKER);
		// Listen for start of tag, get attributes and set them
		// on current marker
		marker.setStartElementListener(new StartElementListener() {
			public void start(final Attributes attributes) {
				segment.clearPoints();
				GeoPoint p;

				final String pointString = attributes.getValue("points");
				final String nameString = attributes.getValue("name");
				String turnString = attributes.getValue("turn");
				final String type = attributes.getValue("type");
				final String walk = attributes.getValue("walk");
				final String length = attributes.getValue("distance");
				final String totalDistance = attributes.getValue("length");
				final String elev = attributes.getValue("elevations");
				final String distances = attributes.getValue("distances");
				final String id = attributes.getValue("itinerary");

				/** Parse segment. **/
				if ("segment".equals(type)) {
					StringBuffer sBuf = new StringBuffer();
					 if (!"unknown".equals(turnString)) {
```

```
                    sBuf.append(Character.toUpperCase(
                            turnString.charAt(0)) + turnString.substring(1));
                    sBuf.append(" at ");
                }
                sBuf.append(nameString);
                sBuf.append(' ');
                if ("1".equals(walk)) {
                    sBuf.append("(dismount)");
                }
                segment.setInstruction(sBuf.toString());

            final String[ ] pointsArray = pointString.split(" ", -1);
            final String[ ] elevations = elev.split(",", -1);
            final String[ ] dists = distances.split(",", -1);

            //Add elevations to the elevation/distance series
            for (int i = 0; i < dists.length; i++) {
                int len = Integer.parseInt(dists[i]);
                int elevation = Integer.parseInt(elevations[i]);
                distance += len;
                route.addElevation(elevation, distance);
            }

            final int len = pointsArray.length;
            for (int i = 0; i < len; i++) {
                final String[ ] point = pointsArray[i].split(",", -1);
                p = new GeoPoint(Convert.asMicroDegrees(Double.parseDouble(point[1])),
                        Convert.asMicroDegrees(Double.parseDouble(point[0])));
                route.addPoint(p);
                segment.addPoint(p);
            }
            segment.setDistance(distance/1000);
            segment.setLength(Integer.parseInt(length));


        } else {
            /** Parse route details. **/
            route.setName(nameString);
            route.setLength(Integer.parseInt(totalDistance));
            route.setItineraryId(Integer.parseInt(id));
        }
    }
});
marker.setEndElementListener(new EndElementListener() {
    public void end() {
        if (segment.getInstruction() != null) {
            route.addSegment(segment.copy());
        }
    }
```

```
            });
            try {
                Xml.parse(this.getInputStream(), Xml.Encoding.UTF_8, root
                        .getContentHandler());
            } catch (Exception e) {
                Log.e(e.getMessage(), "CycleStreets parser - " + feedUrl);
                return null;
            }
            //route.buildTree();
            return route;
        }
}
```

19.5　海拔数据分析

知识点讲解：光盘:视频\知识点\第 19 章\海拔数据分析.avi

为了实时测量不同线路上各个地点的海拔数据，在本实例中使用了 Google Elevation API 技术。Google Elevation API 为开发者提供了查询地球上某位置的海拔数据的简单接口。此外，还可以请求对沿途的海拔数据进行抽样，以便计算沿途路线的海拔变化。本实例实现海拔测量功能的实现文件是 GoogleElevationParser.java，具体实现代码如下所示。

```
public class GoogleElevationParser extends XMLParser implements Parser {
    private double distance;
    private final Route route;

    public GoogleElevationParser(final String feedUrl, final Route route) {
        super(feedUrl);
        this.route = route;
    }
    public Route parse() {
        // turn the stream into a string
        final String result = convertStreamToString(this.getInputStream());
        try {
            //Tranform the string into a json object
            final JSONObject json = new JSONObject(result);
            //Get the results object
            final JSONArray jsonResult = json.getJSONArray("results");
            JSONObject location = jsonResult.getJSONObject(0).getJSONObject("location");
            //Get the first point, use it as a basis for calculating the distance
            //Store incremented distance with retrieved elevation.
            double lastLat = location.getDouble("lat");
            double lastLng = location.getDouble("lng");
            for (int i = 0; i < jsonResult.length(); i++) {
                location = jsonResult.getJSONObject(i).getJSONObject("location");
                final double lat = location.getDouble("lat");
                final double lng = location.getDouble("lng");
                final double elevation = jsonResult.getJSONObject(i).getDouble("elevation");
                distance += pointDiff(lat, lng, lastLat, lastLng);
```

```
                lastLat = lat;
                lastLng = lng;
                route.addElevation(elevation, distance);
            }
        } catch (JSONException e) {
            Log.e(e.getMessage(), "Google JSON Parser - " + feedUrl);
        }
        return route;
    }


    private static String convertStreamToString(final InputStream input) {
        final BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        final StringBuilder sBuf = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sBuf.append(line);
            }
        } catch (IOException e) {
            Log.e(e.getMessage(), "Google parser, stream2string");
        } finally {
            try {
                input.close();
            } catch (IOException e) {
                Log.e(e.getMessage(), "Google parser, stream2string");
            }
        }
        return sBuf.toString();
    }

    private double pointDiff(final double lat, final double lng, final double latA, final double lngA) {
        final double dLat = Math.toRadians(latA - lat);
        final double dLon = Math.toRadians(lngA - lng);
        final double a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
                Math.cos(Math.toRadians(lat)) * Math.cos(Math.toRadians(latA)) *
                Math.sin(dLon / 2) * Math.sin(dLon / 2);
        final double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
        return BikeRouteConsts.EARTH_RADIUS * c * 1000;
    }
}
```

到此为止，本实例的核心功能全部讲解完毕。至于其他模块的具体实现，因与本书中前面的内容有所重复，为了节省篇幅，将不再详细讲解，读者只需阅读本书附带光盘中的源码文件即可。